```
   SSSSSSSSSSS  DDDDDDDDDDD        AAAAAAAAA
   SSSSSSSSSSS  DDDDDDDDDDDD       AAAAAAAAA
   SSSSSSSSSSS  DDDDDDDDDDDD       AAAAAAAAA
SSS             DDD          DDD   AAA         AAA
SSS             DDD          DDD   AAA         AAA
SSS             DDD          DDD   AAA         AAA
SSS             DDD          DDD   AAA         AAA
SSS             DDD          DDD   AAA         AAA
SSS             DDD          DDD   AAA         AAA
   SSSSSSSSS    DDD          DDD   AAA         AAA
   SSSSSSSSS    DDD          DDD   AAA         AAA
   SSSSSSSSS    DDD          DDD   AAA         AAA
            SSS DDD          DDD   AAAAAAAAAAAAAAA
            SSS DDD          DDD   AAAAAAAAAAAAAAA
            SSS DDD          DDD   AAAAAAAAAAAAAAA
            SSS DDD          DDD   AAA         AAA
            SSS DDD          DDD   AAA         AAA
            SSS DDD          DDD   AAA         AAA
SSSSSSSSSSS     DDDDDDDDDDDD       AAA         AAA
SSSSSSSSSSS     DDDDDDDDDDDD       AAA         AAA
SSSSSSSSSSS     DDDDDDDDDDDD       AAA         AAA
```

```
  CCCCCCCC  LL          UU      UU    SSSSSSSS  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
  CCCCCCCC  LL          UU      UU    SSSSSSSS  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
CC          LL          UU      UU  SS            TT      EE          RR      RR
CC          LL          UU      UU  SS            TT      EE          RR      RR
CC          LL          UU      UU  SS            TT      EE          RR      RR
CC          LL          UU      UU  SS            TT      EE          RR      RR
CC          LL          UU      UU    SSSSSS      TT      EEEEEEEE    RRRRRRRR
CC          LL          UU      UU    SSSSSS      TT      EEEEEEEE    RRRRRRRR
CC          LL          UU      UU          SS    TT      EE          RR  RR
CC          LL          UU      UU          SS    TT      EE          RR  RR
CC          LL          UU      UU          SS    TT      EE          RR    RR        ....
CC          LL          UU      UU          SS    TT      EE          RR    RR        ....
  CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU  SSSSSSSS      TT      EEEEEEEEEE  RR      RR      ....
  CCCCCCCC  LLLLLLLLLL  UUUUUUUUUU  SSSSSSSS      TT      EEEEEEEEEE  RR      RR      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  CLUSTER          SHOW CLUSTER INFORMATION
0000     2          .SBTTL  COPYRIGHT NOTICE
0000     3          .IDENT  'V04-000'
0000     4 ;
0000     5 ;****************************************************************************
0000     6 ;*                                                                          *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     9 ;*  ALL RIGHTS RESERVED.                                                    *
0000    10 ;*                                                                          *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    16 ;*  TRANSFERRED.                                                            *
0000    17 ;*                                                                          *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    20 ;*  CORPORATION.                                                            *
0000    21 ;*                                                                          *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    24 ;*                                                                          *
0000    25 ;*                                                                          *
0000    26 ;****************************************************************************
0000    27 ;
```

```
0000     29          .SBTTL  PROGRAM DESCRIPTION
0000     30 ;++
0000     31 ;   FACILITY
0000     32 ;
0000     33 ;       SYSTEM DUMP ANALYZER
0000     34 ;
0000     35 ;   ABSTRACT
0000     36 ;
0000     37 ;       THIS MODULE CONTAINS THE ROUTINES NECESSARY TO DISPLAY THE
0000     38 ;       VAXCLUSTER DATA STRUCTURES.  IT PROVIDES SUPPORT FOR THE COMMANDS,
0000     39 ;       SHOW CLUSTER,SHOW RSPID, SHOW CONNECTIONS, AND SHOW PORTS.
0000     40 ;
0000     41 ;   ENVIRONMENT
0000     42 ;
0000     43 ;       NATIVE MODE, USER MODE
0000     44 ;
0000     45 ;   AUTHOR
0000     46 ;
0000     47 ;       ELLEN M. BATBOUTA, MAY 1984
0000     48 ;
0000     49 ;       V03-001  EMD0110        Ellen M. Batbouta               16-JUL-1984
0000     50 ;                Allocate storage dynamically for the cdl and the rdt since
0000     51 ;                their sizes depend on sysgen parameters.  Display closed
0000     52 ;                cdts only if the /address qualifier is specified on the
0000     53 ;                command.  Also fix a few minor problems with the displays.
0000     54 ;
```

```
0000     56  .sbttl      declarations
0000     57 ;
0000     58 ;       symbol definitions
0000     59 ;
0000     60          $cdldef                 ; Connection Descriptor List (CDL)
0000     61          $cdtdef                 ; SCS Connection Descriptor Table (CDT)
0000     62          $cdrpdef                ; Class Driver Request Packet (CDRP)
0000     63          $clubdef                ; Cluster Block (CLUB)
0000     64          $cludcbdef              ; Cluster Quorum Disk Control Block (CLUDCB)
0000     65          $csbdef                 ; Cluster System Block (CSB)
0000     66          $ddbdef                 ; Device Data Block (DDB)
0000     67          $dyndef                 ; Dynamic Storage Type Definitions
0000     68          $pbdef                  ; Path Block (PB)
0000     69          $pdtdef                 ; Port Descriptor Table (PDT)
0000     70          $rddef                  ; SCS Response Descriptor Format
0000     71          $rdtdef                 ; SCS Response Descriptor Table
0000     72          $sbdef                  ; System Block (SB)
0000     73          $sdirdef                ; SCS Directory Entry (SDIR)
0000     74          $tpadef                 ; TPARSE definitions
0000     75          $ucbdef                 ; Unit Control Block (UCB)
```

```
                     0000     77      .sbttl      storage definitions
                     0000     78 ;
                     0000     79 ;        storage definitions
                     0000     80 ;
                     0000     81
                 00000000     82      .psect      sdadata,noexe,wrt
                     0000     83
00000004             0000     84 cdl:     .blkl   1                       ; to contain address of local cdl
                     0004     85                                          ; Connections Descriptor List (CDL)
                     0004     86 cdl_size:
00000008             0004     87         .blkl   1                       ; to contain size of cdl
                     0008     88
000000A8             0008     89 cdt:     .blkb   cdt$c_length            ; connection descriptor table (CDT)
                     00A8     90
00000250             00A8     91 club:    .blkb   club$c_length           ; Cluster Block (CLUB)
                     0250     92
000002FC             0250     93 csb:     .blkb   csb$c_length            ; Cluster System Block (CSB)
                     02FC     94
00000525             02FC     95 cludcb:  .blkb   cludcb$c_length         ; Cluster Quorum Disk Control Block
                     0525     96
00000609             0525     97 pdt:     .blkb   pdt$c_length            ; Port Descriptor Table (PDT)
                     0609     98
                     0609     99 directory:
00000639             0609    100         .blkb   sdir$c_length           ; SCS directory entry
                     0639    101
0000063D             0639    102 rdt:    .blkl   1                       ; to contain address of local rdt
                     063D    103
                     063D    104 rdt_size:
00000641             063D    105         .blkl   1                       ; to contain size of rdt
                     0641    106 wait_cdrp:
00000000             0641    107         .long   0                       ; cdrp in rdt wait queue
                     0645    108
                     0645    109 sblock:
000006A5             0645    110         .blkb   sb$c_length             ; System Block (SB)
                     06A5    111
000006B5             06A5    112 node:   .blkb   sb$s_nodename           ; node name in system block (SB)
                     06B5    113
000006C5             06B5    114 procname:   .blkb  16                   ; to hold local/remote process name
                     06C5    115
000006D9             06C5    116 driver_name:    .blkb  20               ; driver name
                     06D9    117
000006ED             06D9    118 device_name:    .blkb   20              ; device name
                     06ED    119
                     06ED    120 tim_buffer:
000006F5             06ED    121         .blkl   2                       ; buffer to hold date/time stamp
                     06F5    122 csid::
00000000             06F5    123         .long   0                       ; cluster system id
                     06F9    124
                     06F9    125 cdt_spcfy::
00000000             06F9    126         .long   0                       ; flag to specify if /connection
                     06FD    127                                         ; qualifier was present in command
                     06FD    128
```

```
                        130     .sbttl      read-only data definitions
                        131 ;
                        132 ;       read-only data definitions
                        133 ;
                        134
00000000                135     .psect      cluster,exe,nowrt,long
0000                    136     .default    displacement,long
0000                    137
0000                    138 club_summary:
0000                    139             table   club$v_,<qf_dynvote,qf_vote,quorum,transition>
0028                    140
0028                    141 csb_summary:
0028                    142             table   csb$v_,<long_break,member,removed,qf_same,qf_active>
0058                    143
0058                    144 csb_states:
0058                    145             table   csb$k_,<open,status,reconnect,new,connect,accept,disconnect,-
0058                    146                             reaccept,wait,dead,local>
00B8                    147
00B8                    148 csb_status:
00B8                    149             table   csb$v_,<long_break,member,removed,qf_same,cluster,qf_active,-
00B8                    150                             shutdown,locked,selected,local,status_rcvd,send_status>
0120                    151
0120                    152 fcb_status:
0120                    153             table   clufcb$v_,<active,pending,sync_node,fkb_busy>
0148                    154
0148                    155 club_flags:
0148                    156             table   club$v_,<cluster,qf_active,shutdown,sts_pphase,sts_ph0,-
0148                    157                             sts_ph1b,sts_ph1,sts_ph2,fkb_busy,unlock,no_form,-
0148                    158                             init,backout,lost_cnx,qf_failed_node,qf_vote,-
0148                    159                             qf_newvote,adj_quorum,quorum,transition,qf_dynvote>
01F8                    160
01F8                    161 cludcb_state:
01F8                    162             table   cludcb$v_,<qs_not_ready,qs_ready,qs_active,qs_cluster,qs_vote>
0228                    163
0228                    164 cludcb_flags:
0228                    165             table   cludcb$v_,<qf_tim,qf_rip,qf_wip,qf_error,qf_cspack>
0258                    166
0258                    167 cdt_state:
0258                    168             table   cdt$c_,<closed,listen,open,disc_ack,disc_rec,disc_sent,-
0258                    169                             disc_mtch,con_sent,con_ack,con_rec,accp_sent,rej_sent,-
0258                    170                             vc_fail>
02C8                    171
02C8                    172 cdt_blkstate:
02C8                    173             table   cdt$c_,<con_pend,accp_pend,rej_pend,disc_pend,cr_pend,dcr_pend>
0300                    174
0300                    175 pdt_type:
0300                    176             table   pdt$c_,<pa,pu,pe,ps>
0328                    177
0328                    178 port_char:
0328                    179             table   pdt$v_,<snglhost>
0338                    180
```

```
                     0338   182     .sbttl      show_cluster  --- display structures relevant to vaxclusters
                     0338   183 ;---
                     0338   184 ;
                     0338   185 ;       show_cluster
                     0338   186 ;
                     0338   187 ;       This is the main routine whose purpose is to provide information
                     0338   188 ;       on vaxclusters.  Several structures are displayed.  The order
                     0338   189 ;       is as follows:
                     0338   190 ;                       list of cluster system blocks (CSBs)
                     0338   191 ;                       the cluster block (CLUB)
                     0338   192 ;                       the cluster failover control block (CLUFCB)
                     0338   193 ;                       the cluster quorum disk control block (CLUDCB)
                     0338   194 ;                       display a csb for each node in the cluster
                     0338   195 ;
                     0338   196 ;       Inputs:
                     0338   197 ;
                     0338   198 ;               AP = pointer to TPARSE block
                     0338   199 ;               CSID = cluster system id (CSID)
                     0338   200 ;
                     0338   201 ;       Outputs:
                     0338   202 ;
                     0338   203 ;               Vaxcluster data structures ( as listed above) are shown
                     0338   204 ;---
                     0338   205     .enable lsb
                     0338   206 show_cluster::
             0FFC    0338   207         .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                     033A   208
                     033A   209         subhd   <VAXcluster data structures>            ; set heading
                     0347   210
                     0347   211         getmem  @clu$gl_club,r5                         ; get address of club
          03 50 E8   0357   212         blbs    r0,5$                                   ; branch if able to read it
           0084 31   035A   213         brw     20$                                     ; branch because of error
                     035D   214 5$:
52 000000A8'EF 9E    035D   215         movab   club,r2                                 ; will contain local copy of club
                     0364   216         getmem  (r1),(r2),#club$c_length                ; move club to local storage
          69 50 E9   0375   217         blbc    r0,20$                                  ; check for error
    000006F5'EF B5   0378   218         tstw    csid                                    ; check to see if csid in command
              65 12  037E   219         bneq    locate_csb                              ; display csb of this csid and exit
                     0380   220
              55 DD  0380   221         pushl   r5                                      ; address of club
              52 DD  0382   222         pushl   r2                                      ; pass address of local club
0000043F'EF   01 FB  0384   223         calls   #1,cluster_summary                      ; display list of csb's
                     038B   224
              55 DD  038B   225         pushl   r5                                      ; address of club
              52 DD  038D   226         pushl   r2                                      ; pass address of local club
000005A8'EF   01 FB  038F   227         calls   #1,display_club                         ; display cluster block
                     0396   228
          010C C2 9F 0396   229         pushab  club$b_clufcb(r2)                       ; address of fcb in local storage
          010C C5 9F 039A   230         pushab  club$b_clufcb(r5)                       ; failover control block
0000093D'EF   01 FB  039E   231         calls   #1,display_clufcb                               ; display it
                     03A5   232
          00B4 C2 D5 03A5   233         tstl    club$l_cludcb(r2)                       ; cludcb exists?
              0B 13  03A9   234         beql    6$                                      ; equal, does not exist
          00B4 C2 DD 03AB   235         pushl   club$l_cludcb(r2)                       ; quorum disk control block
000009D6'EF   01 FB  03AF   236         calls   #1,display_cludcb                       ; display it
                     03B6   237 6$:
          54 65 DE   03B6   238         moval   club$l_csbqfl(r5),r4                    ; address of csb queue
```

```
                62   54   D1  03B9   239          cmpl    r4,club$l_csbqfl(r2)            ; check if queue empty
                     23   13  03BC   240          beql    20$                             ; equal, then empty, so exit
                53   62   D0  03BE   241          movl    club$l_csbqfl(r2),r3            ; queue not empty
                     53   DD  03C1   242 10$:      pushl   r3                              ; pass it to routine
        00000B72'EF   01   FB  03C3   243          calls   #1,display_csb                  ; display this csb
                53   00   C0  03CA   244          addl2   #csb$l_sysqfl,r3                ; check for another csb
                          03CD   245          getmem  (r3),r3                         ; read field in queue
                  05 50   E9  03D9   246          blbc    r0,20$                          ; are we able to read it?
                53   54   D1  03DC   247          cmpl    r4,r3                           ; check to see if at end of queue
                     E0   12  03DF   248          bneq    10$                             ; not equal, another csb exists
                50   01   D0  03E1   249 20$:      movl    #1,r0                           ; finished! - return success
                          04  03E4   250          ret
                              03E5   251
                              03E5   252
                              03E5   253
                              03E5   254 locate_csb:
                54   65   DE  03E5   255          moval   club$l_csbqfl(r5),r4            ; start of queue
                62   54   D1  03E8   256          cmpl    r4,club$l_csbqfl(r2)            ; is queue empty
                     F4   13  03EB   257          beql    20$                             ; equal, yes so exit
                53   62   D0  03ED   258          movl    club$l_csbqfl(r2),r3            ; first entry in queue
                              03F0   259 40$:
     56   53  0000004C 8F   C1  03F0   260          addl3   #csb$l_csid,r3,r6               ; point to csid in csb
                          03F8   261          getmem  (r6)                            ; read in csid value
                    DD 50   E9  0401   262          blbc    r0,20$                          ; exit if can not read
        000006F5'EF   51   B1  0404   263          cmpw    r1,csid                         ; right csb?
                     19   13  040B   264          beql    50$                             ; equal, yes so display
                53   00   C0  040D   265          addl2   #csb$l_sysqfl,r3                ; point to next entry in queue
                          0410   266          getmem  (r3),r3                         ; read it in
                  C2 50   E9  041C   267          blbc    r0,20$                          ; exit if not possible
                53   54   D1  041F   268          cmpl    r4,r3                           ; end of queue yet
                     BD   13  0422   269          beql    20$                             ; equal, yes so exit
                     CA   11  0424   270          brb     40$                             ; get next csb
                              0426   271 50$:
                              0426   272          skip    page                            ; next screen
                          53   DD  042D   273          pushl   r3                              ; actual address of csb
        00000B72'EF       01   FB  042F   274          calls   #1,display_csb                  ; display
              000006F5'EF       D4  0436   275          clrl    csid                            ; reinitialize
                        FFA2    31  043C   276          brw     20$                             ; and exit with success
                              043F   277
                              043F   278          .dsabl  lsb
```

```
                                J 15
                     043F    280      .sbttl      cluster_summary --- summary sheet for the club and csbs
                     043F    281 ;---
                     043F    282 ;
                     043F    283 ;       cluster_summary
                     043F    284 ;
                     043F    285 ;       This routine outputs a brief summary of the cluster block (CLUB)
                     043F    286 ;       and of each cluster system block (CSB).  There exists
                     043F    287 ;       one csb per node in the cluster and one club for the cluster.
                     043F    288 ;
                     043F    289 ;       Inputs:
                     043F    290 ;
                     043F    291 ;               4(ap) = address of club in local storage
                     043F    292 ;               8(ap) = actual address of club
                     043F    293 ;
                     043F    294 ;       Outputs:
                     043F    295 ;
                     043F    296 ;
                     043F    297 ;---
                     043F    298 cluster_summary:
                OFFC 043F    299      .word     ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                     0441    300
      54  04 AC  D0 0441    301      movl      4(ap),r4                        ; club in local storage
                     0445    302 ;
                     0445    303 ; First display a few important fields in the cluster block (CLUB)
                     0445    304 ;
                     0445    305      skip      page
                     044C    306      print     0,<!_                --- VAXcluster Summary --->
                     0459    307      skip      1
                     0462    308      print     0,<!_ Quorum   Votes   Quorum Disk Votes   Status Summary>
                     046F    309      print     0,<!_ ------   -----   ----------------   -------------->
                     047C    310
                     047C    311      alloc     80,r2                           ; allocate output buffer
        7E    1C A4  D0 048E    312      movl      club$l_flags(r4),-(sp)  ; bit mask to translate
            FB6A CF  9F 0492    313      pushab    club_summary                    ; address of definition table
00000000'EF    02  FB 0496    314      calls     #2,translate_bits               ; translate bits to names
                     049D    315
              5E  DD 049D    316      pushl     sp                              ; address of string descriptor
      7E   00AE C4  3C 049F    317      movzwl    club$w_qdvotes(r4),-(sp)              ; quorum disk votes
        7E    22 A4  3C 04A4    318      movzwl    club$w_votes(r4),-(sp)  ; cluster votes
        7E    20 A4  3C 04A8    319      movzwl    club$w_quorum(r4),-(sp) ; cluster quorum
                     04AC    320      print     4,<!_    !4<!UW!>    !4<!UW!>          !4<!UW!>          !AS>
5E   00000058 8F  C0 04B9    321      addl2     #88,sp                          ; clean up stack
                     04C0    322 ;
                     04C0    323 ; Now to actually display a list of csb's and a little information about
                     04C0    324 ; each one. (A little knowledge never hurt anyone, right?)
                     04C0    325 ;
                     04C0    326      skip      1
                     04C9    327      print     0,<!_!_!_--- CSB list --->
                     04D6    328      skip      1
                     04DF    329      print     0,<Address   Node    CSID      Votes   State     Status>
                     04EC    330      print     0,<-------   ----    ----      -----   -----     ------>
                     04F9    331      skip      1
                     0502    332 ;
                     0502    333 ; Header information complete - now time to loop through the queue of csb's
                     0502    334 ; in the cluster block (club)
                     0502    335 ;
                     0502    336      assume    club$l_csbqfl eq 0
```

```
                64    08 AC    D1   0502   337      cmpl    8(ap),club$l_csbqfl(r4)                 ; check for empty queue
                         03    12   0506   338      bneq    20$                                     ; not equal, entry in queue
                       0099    31   0508   339      brw     done                                    ; otherwise, this display is done
                                    050B   340 20$:
                   56    64    D0   050B   341      movl    club$l_csbqfl(r4),r6                    ; get address of csb
                                    050E   342 Loop:
            57 00000250'EF       DE   050E   343      moval   csb,r7                                  ; local storage for csb
                                    0515   344      getmem  (r6),(r7),#csb$c_length                 ; read entire csb
                      7B 50      E9   0526   345      blbc    r0,done                                 ; if not able to read, exit
                                    0529   346
                                    0529   347      alloc   80                                      ; alloc buffer for translation
                   7E    60 A7    D0   0538   348      movl    csb$l_status(r7),-(sp)                  ; bit mask to translate
                      FAE8 CF       9F   053C   349      pushab  csb_summary                             ; bit definition table
            00000000'EF    02     FB   0540   350      calls   #2,translate_bits                       ; translate bits to names
                         5E       DD   0547   351      pushl   sp                                      ; names for status bits
                                    0549   352
                   52    43 A7    9A   0549   353      movzbl  csb$b_state(r7),r2                      ; bit mask to translate
                53    FB07 CF       9E   054D   354      movab   csb_states,r3                           ; state translation table
                 00000000'GF       16   0552   355      jsb     g^translate_address                     ; translate value to names
                         02       13   0558   356      beql    10$                                     ; branch if translation failed
                         50       DD   055A   357      pushl   r0                                      ; names for states
                                    055C   358 10$:
                   7E    50 A7    3C   055C   359      movzwl  csb$w_votes(r7),-(sp)                   ; votes held by node
                         4C A7    DD   0560   360      pushl   csb$l_csid(r7)                          ; Cluster System Id
                                    0563   361
                   52 000006A5'EF       9E   0563   362      movab   node,r2
         53    68 A7 00000044 8F       C1   056A   363      addl3   #sb$t_nodename, csb$l_sb(r7),r3  ; point to nodename
                                    0573   364      getmem  (r3),(r2),#sb$s_nodename                ; read in nodename
                         52       DD   0580   365      pushl   r2                                      ; node
                         56       DD   0582   366      pushl   r6                                      ; address of csb
                                    0584   367      print   6,<!XL  !6<!AC!>  !XL    !3<!UW!>!10<   !AC!> !AS>
                  5E 00000058 8F       C0   0591   368      addl2   #88,sp                                 ; clean up stack
                                    0598   369
                08 AC    67       D1   0598   370      cmpl    csb$l_sysqfl(r7),8(ap)                  ; check for end of csbs
                         06       13   059C   371      beql    done                                    ; equal, at end
                   56    67       D0   059E   372      movl    csb$l_sysqfl(r7),r6                     ; address of next csb
                      FF6A       31   05A1   373      brw     loop                                    ; loop to display
                                    05A4   374
                                    05A4   375 done:
                   50    01       D0   05A4   376      movl    #1,r0
                                    04   05A7   377      ret
```

```
                        05A8   379      .sbttl      display_club --- display cluster block (CLUB)
                        05A8   380 ;---
                        05A8   381 ;
                        05A8   382 ;       display_club
                        05A8   383 ;
                        05A8   384 ;       This routine displays the cluster block.  There exists
                        05A8   385 ;       one club per cluster.
                        05A8   386 ;
                        05A8   387 ;       Inputs:
                        05A8   388 ;
                        05A8   389 ;               4(ap) = address of club in local storage
                        05A8   390 ;               8(ap) = actual address of club
                        05A8   391 ;
                        05A8   392 ;       Outputs:
                        05A8   393 ;               The CLUB is displayed.
                        05A8   394 ;
                        05A8   395 ;---
                        05A8   396 display_club:
                   0FFC 05A8   397      .word       ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        05AA   398
                        05AA   399      ensure      20                          ; make sure at least 20 lines on screen
              54  04 AC  D0   05C2   400      movl        4(ap),r4                    ; club in local storage
                  08 AC  DD   05C6   401      pushl       8(ap)                       ; actual address of club
                        05C9   402      print       1,<!_!_ --- Cluster Block (CLUB) !XL --->
                        05D6   403      skip        1
                        05DF   404      alloc       80,r5                                   ; 80 byte output bu.fer
                    1C A4  DD   05F1   405      pushl       club$l_flags(r4)                ; flags in club
                FB50 CF  9F   05F4   406      pushab      club_flags                      ; bit definition table
     00000000'EF    02  FB   05F8   407      calls       #2,translate_bits               ; translate bits to names
                    55  DD   05FF   408      pushl       r5                              ; names defining flags
                    1C A4  DD   0601   409      pushl       club$l_flags(r4)                ; flags in club
                        0604   410      print       2,<Flags: !XL !AS>              ; display translated flags
                        0611   411      skip        1
           5E   00000058 8F   C0   061A   412      addl2       #88,sp                          ; clean up stack
                        0621   413      print_columns -
                        0621   414                  (r4),8(ap),-
                        0621   415                  club_col_1,club_col_2           ; display the club
                        0639   416      status      success
                     04 0640   417      ret
```

```
                                                M 15
0641    419     .sbttl      cluster block data block tables & action routines
0641    420 ;
0641    421 ;   PRINT_COLUMNS table for CLUB displays
0641    422 ;
0641    423
0641    424 club_fao_6bytes:
0641    425         string  <!#* !XW!XL>
0653    426
0653    427 club_2words:
0653    428         string  <!10UW/!UW>
0664    429
0664    430 club_col_1:
0664    431         column_list     -
0664    432                 club$, 21, 12, 4, <-
0664    433                 <<Quorum/Votes>,quor_vote,0>,-
0664    434                 <<Quorum Disk Votes>,w_qdvotes,uw>,-
0664    435                 <<Nodes>,w_nodes,uw>,-
0664    436                 <<Quorum Disk>,t_qdname,ac,15,18>,-
0664    437                 <<Found Node SYSID>,club_6bytes,club$b_fsysid>,-
0664    438                 <<Founding Time>,date_routine,club$q_ftime>,-
0664    439                 <<>,time_routine,club$q_ftime>,-
0664    440                 <<Index of next CSID>,w_next_csid,xw>,-
0664    441                 <<Quorum Disk Cntrl Block>,l_cludcb,xl,25,8>,-
0664    442                 <<Timer Entry Address>,l_tqe,xl>,-
0664    443                 <<CSP Queue>,l_cspfl,q2>,-
0664    444                 <<Transaction code>,trans_byte,club$b_cur_code>,-
0664    445                 <<Transaction Phase>,trans_byte,club$b_cur_phase>,-
0664    446                 <<Message Count>,trans_word,club$w_msgcnt>,-
0664    447                 >
0754    448
0754    449
0754    450 club_col_2:
0754    451         column_list     -
0754    452                 club$, 21, 12, 0, < -
0754    453                 <<Last transaction code>,b_lst_code,xb>,-
0754    454                 <<Last trans. number>,l_lst_xtn,ul>,-
0754    455                 <<Last coordinator CSID>,l_lst_coord,xl>,-
0754    456                 <<Last time stamp>,date_routine,club$q_lst_time>,-
0754    457                 <<>,time_routine,club$q_lst_time>,-
0754    458                 <<Largest trans. id>,l_max_xtn,xl>,-
0754    459                 <<Resource Alloc. retry>,l_retrycnt,ul>,-
0754    460                 <<Figure of Merit>,l_fmerit,xl>,-
0754    461                 <<Member State Seq. Num.>,w_memseq,xw>,-
0754    462                 <<Foreign Cluster>,l_foreign_cluster,xl>,-
0754    463                 <<Curr. coord. CSID>,trans_long,club$l_cur_coord>,-
0754    464                 <<Current trans. number>,trans_long,club$l_cur_xtn>,-
0754    465                 <<Curr. time-stamp>,curr_date,club$q_cur_time>,-
0754    466                 <<>,curr_time,club$q_cur_time>,-
0754    467                 >
0844    468
0844    469 ;
0844    470 ;   The following are all PRINT_COLUMNS action routines for the show
0844    471 ;   cluster block displays.
0844    472 ;
0844    473 ;   Action Routine Inputs:
0844    474 ;
0844    475 ;       R2      value from the COLUMN_LIST entry
```

```
                                0844   476 ;       R5      size of value section for this item
                                0844   477 ;       R7      address of a descriptor for a scratch string in
                                0844   478 ;               which the FAO converted value is to be returned
                                0844   479 ;       R11     base address of the local CLUB copy
                                0844   480 ;
                                0844   481 ;   Action Routine Outputs:
                                0844   482 ;
                                0844   483 ;       R0      status
                                0844   484 ;               lbs ==> use this entry
                                0844   485 ;               lbc ==> skip this entry
                                0844   486 ;       R1-R5   scratch
                                0844   487 ;               all other registers must be preserved
                                0844   488 ;
                                0844   489 ;*******
                                0844   490 quor_vote:
                  52    20 AB   3C  0844   491     movzwl      club$w_quorum(r11),r2           ; display quorum value with
                  53    22 AB   3C  0848   492     movzwl      club$w_votes(r11),r3            ; the value for votes
                                084C   493     $fao_s      -
                                084C   494                 ctrstr = club_2words,-          ; two values as requested
                                084C   495                 outbuf=(r7),-
                                084C   496                 outlen=(r7),-
                                084C   497                 p1=r2,-
                                084C   498                 p2=r3
                             05 085F   499     rsb
                                0860   500
                                0860   501 ;*******
                                0860   502 club_6bytes:
                  53    5B    52  C1  0860   503     addl3       r2,r11,r3       ; locate storage of interest
                        55    0C  C2  0864   504     subl        #12,r5          ; get size of filler field
                                0867   505     $fao_s      -
                                0867   506                 ctrstr=club_fao_6bytes,-
                                0867   507                 outbuf=(r7),-
                                0867   508                 outlen=(r7),-
                                0867   509                 p1=r5,-
                                0867   510                 p2=4(r3),-
                                0867   511                 p3=(r3)
                             05 087D   512     rsb
                                087E   513
                                087E   514 ;*******
                                087E   515 date_routine:
                     53 5B    52  C1  087E   516     addl3       r2,r11,r3       ; locate area of interest
                                0882   517
                000006ED'EF     63  7D  0882   518     movq        (r3),tim_buffer ; move into buffer
                                0889   519     alloc       11,r4           ; allocate space for date
                                0893   520     $asctim_s   timadr=tim_buffer,-
                                0893   521                 timbuf=(r4)     ; convert value to ascii
                        52      54  D0  08A6   522     movl        r4,r2           ; pass address of descriptor in r2
                                08A9   523     do_column_entry     as      ; display date
                        5E      14  C0  08B2   524     addl        #20,sp          ; clean up the stack
                             05 08B5   525     rsb
                                08B6   526
                                08B6   527 ;*******
                                08B6   528 time_routine:
                     53 5B    52  C1  08B6   529     addl3       r2,r11,r3       ; locate area of interest
                000006ED'EF     63  7D  08BA   530     movq        (r3),tim_buffer ; move into buffer
                                08C1   531     alloc       24,r4           ; allocate space for date/time
                                08CB   532     $asctim_s   timadr=tim_buffer,-
```

```
                     08CB   533                     timbuf=(r4)     ; convert to ascii
          64   09    B0   08DE   534     movw        #9,(r4)         ; only display time - adjust length accordingly
       04 A4   0B    C0   08E1   535     addl2       #11,4(r4)       ; adjust address to point to time
          52   54    D0   08E5   536     movl        r4,r2           ;
                     08E8   537     do_column_entry     as          ; display time
             5E   20 C0   08F1   538     addl2       #32,sp          ; clean up the stack
                  05     08F4   539     rsb
                     08F5   540
                     08F5   541 ;*******
                     08F5   542 curr_date:
    03 1C AB    1D   E1   08F5   543     bbc         #club$v_transition,club$l_flags(r11),10$
                     08FA   544                                     ; if transition in progress, this field is of
                     08FA   545                                     ; interest to us, so display.
                81 AF    17   08FA   546     jmp         date_routine
                     08FD   547 10$:
                  05     08FD   548     rsb
                     08FE   549
                     08FE   550 ;*******
                     08FE   551 curr_time:
    03 1C AB    1D   E1   08FE   552     bbc         #club$v_transition,club$l_flags(r11),10$
                     0903   553                                     ; if transition in progress, this field is of
                     0903   554                                     ; interest to us, so display
                B0 AF    17   0903   555     jmp         time_routine
                     0906   556 10$:
                  05     0906   557     rsb
                     0907   558
                     0907   559 ;********
                     0907   560 trans_long:
    0C 1C AB    1D   E1   0907   561     bbc         #club$v_transition,club$l_flags(r11),10$
                     090C   562                                     ; if transition in progress, this field is of
                     090C   563                                     ; interest to us, so display.
             52   5B   C0   090C   564     addl        r11,r2          ; locate cell to return
                     090F   565     do_column_entry     xl,jmp
                     0918   566 10$:
                  05     0918   567     rsb
                     0919   568
                     0919   569 ;********
                     0919   570 trans_word:
    0C 1C AB    1D   E1   0919   571     bbc         #club$v_transition,club$l_flags(r11),10$
                     091E   572                                     ; if transition in progress, this field is of
                     091E   573                                     ; interest to us, so display.
             52   5B   C0   091E   574     addl        r11,r2          ; locate cell to return
                     0921   575     do_column_entry     uw,jmp
                     092A   576 10$:
                  05     092A   577     rsb
                     092B   578
                     092B   579 ;********
                     092B   580 trans_byte:
    0C 1C AB    1D   E1   092B   581     bbc         #club$v_transition,club$l_flags(r11),10$
                     0930   582                                     ; if transition in progress, this field is of
                     0930   583                                     ; interest to us, so display.
             52   5B   C0   0930   584     addl        r11,r2          ; locate cell to return
                     0933   585     do_column_entry     ub,jmp
                     093C   586 10$:
                  05     093C   587     rsb
```

```
                                093D   589      .sbttl     display_clufcb --- display cluster failover control block(CLUFCB)
                                093D   590 ;---
                                093D   591 ;
                                093D   592 ;       display_clufcb
                                093D   593 ;
                                093D   594 ;       This routine displays the cluster failover control block which is
                                093D   595 ;       a subblock of the clud that is used to sequence failover actions
                                093D   596 ;       in a cluster
                                093D   597 ;
                                093D   598 ;       Inputs:
                                093D   599 ;
                                093D   600 ;               4(ap) = actual address of cluster failover control block
                                093D   601 ;               8(ap) = address of cluster fcb in local storage
                                093D   602 ;
                                093D   603 ;       Outputs:
                                093D   604 ;               The Cluster failover control block is displayed.
                                093D   605 ;
                                093D   606 ;---
                                093D   607 display_clufcb:
                          0FFC  093D   608     .word      ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                093F   609
                                093F   610     ensure     20                          ; make sure at least 20 lines on screen
                    54  08 AC   D0   0957   611     movl       8(ap),r4                    ; cluster fcb in local storage
                        04 AC   DD   095B   612     pushl      4(ap)                       ; actual address of cluster fcb
                                095E   613     print      1,<!_ --- Cluster Failover Control Block (CLUFCB) !XL --->
                                096B   614     skip       1
                                0974   615     alloc      80,r5                               ; 80 byte output buffer
                      20 A4     DD   0986   616     pushl      clufcb$l_status(r4)         ; status
                  F793 CF       9F   0989   617     pushab     fcb_status                  ; bit definition table
      00000000'EF     02       FB   098D   618     calls      #2,translate_bits           ; translate bits to names
                      55       DD   0994   619     pushl      r5                          ; names defining status bits
                      20 A4    DD   0996   620     pushl      clufcb$l_status(r4)         ; status field
                                0999   621     print      2,<Flags: !XL !AS>          ; display translated status
                                09A6   622     skip       1
          5E    00000058 8F     C0   09AF   623     addl2      #88,sp                      ; clean up stack
                                09B6   624     print_columns -
                                09B6   625                (r4),4(ap),-
                                09B6   626                fcb_col_1,fcb_col_2         ; display the cluster fcb
                                09CE   627     status     success
                          04    09D5   628     ret
```

```
                                09D6   630      .sbttl     display_cludcb --- display cluster quorum disk control block
                                09D6   631 ;---
                                09D6   632 ;
                                09D6   633 ;       display_cludcb
                                09D6   634 ;
                                09D6   635 ;
                                09D6   636 ;       Inputs:
                                09D6   637 ;
                                09D6   638 ;              4(ap) = actual address of cluster quorum disk control block
                                09D6   639 ;
                                09D6   640 ;       Outputs:
                                09D6   641 ;              The Cluster quorum disk control block is displayed.
                                09D6   642 ;
                                09D6   643 ;---
                                09D6   644 display_cludcb:
                          0FFC  09D6   645      .word      ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                09D8   646
                                09D8   647      skip       3
                55   04 AC     D0   09E1   648      movl       4(ap),r5                ; actual address of cluster dcb
                        55     DD   09E5   649      pushl      r5
                                09E7   650      print      1,<!_ --- Cluster Quorum Disk Control Block (CLUDCB) !XL --->
                                09F4   651      skip       1
        54   000002FC'EF       9E   09FD   652      movab      cludcb,r4
                                0A04   653      getmem     (r5),(r4),#cludcb$c_length      ; read into local storage
                   01 50       E8   0A15   654      blbs       r0,20$                          ; branch if able to read
                               05   0A18   655      rsb                                        ; else, exit
                                0A19   656 20$:
                                0A19   657      alloc      80,r6                           ; 80 byte output buffer
             7E    20 A4       3C   0A2B   658      movzwl     cludcb$w_state(r4),-(sp)        ; status
                F7C5 CF       9F   0A2F   659      pushab     cludcb_state                    ; bit definition table
     00000000'EF     02       FB   0A33   660      calls      #2,translate_bits               ; translate bits to names
                     56       DD   0A3A   661      pushl      r6                              ; names defining state bits
             7E    20 A4       3C   0A3C   662      movzwl     cludcb$w_state(r4),-(sp)        ; state field
                               0A40   663      print      2,<State: !XW !AS>              ; display translated state
             66    50 8F       9A   0A4D   664      movzbl     #80,(r6)                        ; reinitialize buffer
             7E    22 A4       3C   0A51   665      movzwl     cludcb$w_flags(r4),-(sp)        ; translate flags now
                F7CF CF       9F   0A55   666      pushab     cludcb_flags                    ; bit definition table
     00000000'EF     02       FB   0A59   667      calls      #2,translate_bits               ; translate to names
                     56       DD   0A60   668      pushl      r6                              ; names defining flags
             7E    22 A4       3C   0A62   669      movzwl     cludcb$w_flags(r4),-(sp)        ; flags field
                               0A66   670      print      2,<Flags: !XW !AS>              ; display translated flags
                               0A73   671      skip       1
     5E    00000058 8F       C0   0A7C   672      addl2      #88,sp                          ; clean up stack
                               0A83   673      print_columns -
                               0A83   674                 (r4),r5,-
                               0A83   675                 dcb_col_1,dcb_col_2             ; display the cluster fcb
                               0A9A   676
                               0A9A   677      status     success
                          04   0AA1   678      ret
```

```
                        .sbttl      cluster failover control block tables & action routines
                    ;
                    ;   PRINT_COLUMNS table for CLUFCB displays
                    ;
                    fcb_col_1:
                            column_list     -
                                    clufcb$, 21, 12, 4, < -
                                    <<Failover Step Index>,l_step,xl>,-
                                    <<Failover Instance ID>,l_id,xl>,-
                                    >

                    fcb_col_2:
                            column_list     -
                                    clufcb$, 21, 12, 4, < -
                                    <<CSB of Synchr. System>,l_sync_csb,xl>,-
                                    >
```

```
                        ;
                        ;  PRINT_COLUMNS table for CLUDCB displays
                        ;
                        dcb_col_1:
                                column_list     -
                                        cludcb$, 21, 12, 4, < -
                                        <<Iteration Counter>,b_counter,ub>,-
                                        <<Activity Counter>,l_act_count,ul>,-
                                        <<Quorum file LBN>,l_qflbn,xl>,-
                                        >

                        dcb_col_2:
                                column_list     -
                                        cludcb$, 21, 12, 0, < -
                                        <<          UCB address>,l_ucb,xl>,-
                                        <<          TQE address>,l_tqe,xl>,-
                                        <<          IRP address>,l_irp,xl>,-
                                        >
```

```
                                0B72   717      .sbttl     display_csb --- display cluster system block (CSB)
                                0B72   718 ;---
                                0B72   719 ;
                                0B72   720 ;       display_csb
                                0B72   721 ;
                                0B72   722 ;
                                0B72   723 ;       Inputs:
                                0B72   724 ;
                                0B72   725 ;               4(ap) = actual address of cluster system block
                                0B72   726 ;
                                0B72   727 ;       Outputs:
                                0B72   728 ;               The Cluster system block is displayed.
                                0B72   729 ;               All registers are preserved.
                                0B72   730 ;
                                0B72   731 ;---
                                0B72   732 display_csb:
                          0FFC  0B72   733      .word      ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                0B74   734
                                0B74   735      ensure     20
                      04 AC     DD 0B8C   736      pushl      4(ap)                   ; actual address of csb
                                0B8F   737      skip       1
           54     00000250'EF   9E 0B98   738      movab      csb,r4                  ; buffer to hold contents of csb locally
                                0B9F   739      getmem     @4(ap),(r4),#csb$c_length       ; read into local storage
                      01 50     E8 0BB1   740      blbs       r0,10$                          ; branch if able to read
                                05 0BB4   741      rsb                                        ; else, exit
                                0BB5   742 10$:
     57   68 A4     00000044 8F   C1 0BB5   743      addl3      #sb$t_nodename,csb$l_sb(r4),r7  ; address of system block
                                0BBE   744      getmem     (r7),node,#sb$s_nodename        ; read node name
                   000006A5'EF   DF 0BCF   745      pushal     node                            ; address of ASCIC string
                                0BD5   746      print      1,<!_ --- !AC Cluster System Block (CSB) !XL --->
                                0BE2   747      skip       1
                                0BEB   748
                                0BEB   749 make_csb_symbols:
                                0BEB   750      make_symbol CSB, 4(ap)
                                0C01   751      make_symbol CDT, csb$l_cdt(r4)
                                0C17   752      make_symbol PDT, csb$l_pdt(r4)
                                0C2D   753      make_symbol SB, csb$l_sb(r4)
                                0C43   754
        52     43 A4     9A 0C43   755      movzbl     csb$b_state(r4),r2              ; state field
     53   F40D CF     9E 0C47   756      movab      csb_states,r3                   ; bit definition table
      00000000'GF     16 0C4C   757      jsb        g^translate_address             ; translate bits to names
                13   13 0C52   758      beql       notrans                         ; equal, unable to translate
                50   DD 0C54   759      pushl      r0                              ; names defining state bits
        7E     43 A4     9A 0C56   760      movzbl     csb$b_state(r4),-(sp)           ; state field
                                0C5A   761      print      2,<State: !XB !AC>              ; display translated state
                                0C67   762 notrans:
                                0C67   763      alloc      80,r6                           ; allocate buffer
                60 A4     DD 0C79   764      pushl      csb$l_status(r4)                ; translate status now
              F438 CF     9F 0C7C   765      pushab     csb_status                      ; bit definition table
00000000'EF     02     FB 0C80   766      calls      #2,translate_bits               ; translate to names
                56   DD 0C87   767      pushl      r6                              ; names defining status
                60 A4     DD 0C89   768      pushl      csb$l_status(r4)                ; status field
                                0C8C   769      print      2,<Flags: !XL !AS>              ; display translated status
                                0C99   770      skip       1
5E     00000058 8F     C0 0CA2   771      addl2      #88,sp                          ; clean up stack
                                0CA9   772      print_columns -
                                0CA9   773                 (r4),4(ap),-
```

```
                0CA9    774                     csb_col_1,csb_col_2,-
                0CA9    775                     csb_col_3                       ; display the csb
                0CC7    776
                0CC7    777     status          success
          04    0CCE    778     ret
                0CCF    779
```

```
                        0CCF    781     .sbttl      cluster system block tables & action routines
                        0CCF    782 ;
                        0CCF    783 ;   PRINT_COLUMNS table for CSB displays
                        0CCF    784 ;
                        0CCF    785
                        0CCF    786 csb_2words:
                        0CCF    787     string      <!6UW/!UW>
                        0CDF    788
                        0CDF    789 csb_2bytes:
                        0CDF    790     string      <!5UB/!UB>
                        0CEF    791
                        0CEF    792 csb_col_1:
                        0CEF    793             column_list     -
                        0CEF    794                     csb$, 14, 8, 4, < -
                        0CEF    795                     <<Quorum/Votes>,csbquor_votes,0>,-
                        0CEF    796                     <<Quor. Disk Vote>,w_qdvotes,uw,15,7>,-
                        0CEF    797                     <<CSID>,l_csid,xl>,-
                        0CEF    798                     <<Eco/Version>,eco_vers,0>,-
                        0CEF    799                     <<Reconn. time>,l_timeout,xl>,-
                        0CEF    800                     <<Ref. count>,b_ref_cnt,ub>,-
                        0CEF    801                     <<Ref. time>,date_routine,csb$q_reftime,11,11>,-
                        0CEF    802                     <<>,time_routine,csb$q_reftime,11,11>,-
                        0CEF    803                     >
                        0D7F    804
                        0D7F    805 csb_col_2:
                        0D7F    806             column_list     -
                        0D7F    807                     csb$, 16, 8, 4, < -
                        0D7F    808                     <<Next seq. number>,w_sendseqnm,xw>,-
                        0D7F    809                     <<Last seq num rcvd>,w_rcvdseqnm,xw,17,7>,-
                        0D7F    810                     <<Last ack. seq num>,w_ackrseqnm,xw,17,7>,-
                        0D7F    811                     <<Unacked messages>,b_unackedmsgs,ub>,-
                        0D7F    812                     <<Ack limit>,b_remacklim,ub,18,6>,-
                        0D7F    813                     <<Incarnation>,date_routine,csb$q_swincarn,13,11>,-
                        0D7F    814                     <<>,time_routine,csb$q_swincarn,13,11>,-
                        0D7F    815                     <<Lock mgr dir wgt>,w_lckdirwt,uw>,-
                        0D7F    816                     >
                        0E0F    817
                        0E0F    818 csb_col_3:
                        0E0F    819             column_list     -
                        0E0F    820                     csb$, 16, 8, 0, < -
                        0E0F    821                     <<Send queue>,l_sentqfl,xl>,-
                        0E0F    822                     <<Resend queue>,l_resendqfl,xl>,-
                        0E0F    823                     <<Block xfer Q.>,l_partnerqfl,q2>,-
                        0E0F    824                     <<CDT address>,l_cdt,xl>,-
                        0E0F    825                     <<PDT address>,l_pdt,xl>,-
                        0E0F    826                     <<TQE address>,l_tqe,xl>,-
                        0E0F    827                     <<SB address>, l_sb,xl>,-
                        0E0F    828                     <<Current CDRP>,l_currcdrp,xl>,-
                        0E0F    829                     >
                        0E9F    830 ;
                        0E9F    831 ;   The following are all PRINT_COLUMNS action routines for the show
                        0E9F    832 ;   cluster block displays.
                        0E9F    833 ;
                        0E9F    834 ;   Action Routine Inputs:
                        0E9F    835 ;
                        0E9F    836 ;       R2      value from the COLUMN_LIST entry
                        0E9F    837 ;       R5      size of value section for this item
```

```
                     0E9F    838 ;       R7      address of a descriptor for a scratch string in
                     0E9F    839 ;               which the FAO converted value is to be returned
                     0E9F    840 ;       R11     base address of the local CLUB copy
                     0E9F    841 ;
                     0E9F    842 ;   Action Routine Outputs:
                     0E9F    843 ;
                     0E9F    844 ;       R0      status
                     0E9F    845 ;               lbs ==> use this entry
                     0E9F    846 ;               lbc ==> skip this entry
                     0E9F    847 ;       R1-R5   scratch
                     0E9F    848 ;               all other registers must be preserved
                     0E9F    849 ;
                     0E9F    850
                     0E9F    851 ;*******
                     0E9F    852 csbquor_votes:
52   52 AB   3C      0E9F    853     movzwl      csb$w_quorum(r11),r2            ; display quorum value with
53   50 AB   3C      0EA3    854     movzwl      csb$w_votes(r11),r3             ; the value for votes
                     0EA7    855     $fao_s      -
                     0EA7    856                 ctrstr = csb_2words,-           ; two values as requested
                     0EA7    857                 outbuf=(r7),-
                     0EA7    858                 outlen=(r7),-
                     0EA7    859                 p1=r2,-
                     0EA7    860                 p2=r3
                05   0EBA    861     rsb
                     0EBB    862 ;*******
                     0EBB    863 eco_vers:
52   40 AB   9A      0EBB    864     movzbl      csb$b_ecolvl(r11),r2            ; display eco level with
53   41 AB   9A      0EBF    865     movzbl      csb$b_vernum(r11),r3            ; the version number
                     0EC3    866     $fao_s      -
                     0EC3    867                 ctrstr = csb_2bytes,-   ; two values as requested
                     0EC3    868                 outbuf=(r7),-
                     0EC3    869                 outlen=(r7),-
                     0EC3    870                 p1=r2,-
                     0EC3    871                 p2=r3
             05      0ED6    872     rsb
                     0ED7    873
```

```
                        0ED7   875      .sbttl      show_scs --- display system communications(SCS) data structures
                        0ED7   876 ;---
                        0ED7   877 ;
                        0ED7   878 ;       show_scs
                        0ED7   879 ;
                        0ED7   880 ;       This is the main routine whose purpose is to provide information
                        0ED7   881 ;       on vaxclusters which are related to system communications (SCS).
                        0ED7   882 ;       Several structures are displayed.  The order
                        0ED7   883 ;       is as follows:
                        0ED7   884 ;                       summary page
                        0ED7   885 ;                       each system block and all of its path blocks
                        0ED7   886 ;
                        0ED7   887 ;       Inputs:
                        0ED7   888 ;
                        0ED7   889 ;               AP = pointer to TPARSE block
                        0ED7   890 ;
                        0ED7   891 ;       Outputs:
                        0ED7   892 ;
                        0ED7   893 ;               SCS data structures ( as mentioned above) are shown
                        0ED7   894 ;               All registers are preserved.
                        0ED7   895 ;
                        0ED7   896 ;---
                        0ED7   897      .enable lsb
                        0ED7   898 show_scs::
                0FFC    0ED7   899      .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        0ED9   900
                        0ED9   901      subhd   <VAXcluster data structures>            ; set heading
                        0EE6   902      skip    page
00000F06'EF   00   FB   0EED   903      calls   #0,scs_summary                         ; summary page
                        0EF4   904
                        0EF4   905      skip page
00001091'EF   00   FB   0EFB   906      calls   #0,display_sb_pbs                       ; system and path blocks
                        0F02   907
              50   01   D0   0F02   908      movl    #1,r0                                   ; return success
                        04   0F05   909      ret
                             0F06   910      .dsabl  lsb
```

```
                        0F06   912     .sbttl      scs_summary --- display system communications(SCS) summary
                        0F06   913 ;---
                        0F06   914 ;
                        0F06   915 ;       show_scs
                        0F06   916 ;
                        0F06   917 ;       This is a coroutine whose purpose is display a summary page.
                        0F06   918 ;       The summary page is divided into 2 parts.  The first half
                        0F06   919 ;       displays a list of the local processes that are known to SCS.
                        0F06   920 ;       The second half displays a brief description of the systems
                        0F06   921 ;       in the cluster and the number of paths each has.
                        0F06   922 ;
                        0F06   923 ;       Inputs:
                        0F06   924 ;
                        0F06   925 ;               none
                        0F06   926 ;
                        0F06   927 ;       Outputs:
                        0F06   928 ;
                        0F06   929 ;               SCS data structures (as mentioned above) are shown
                        0F06   930 ;               All registers are preserved.
                        0F06   931 ;
                        0F06   932 ;---
                        0F06   933             .enabl  lsb
                        0F06   934 scs_summary:
                0FFC    0F06   935             .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        0F08   936
                        0F08   937             skip    1
                        0F11   938             print   0,<!_   --- SCS Listening Process Directory --->
                        0F1E   939             skip    1
                        0F27   940             print   0,<Entry Address     Connection ID     Process Name        Informati
                        0F34   941             print   0,<--------------    -------------     ------------        ---------
                        0F41   942             skip    1
                        0F4A   943
                        0F4A   944             getmem  @scs$gq_direct,r6                       ; get address of 1st entry
                        0F5A   945             retiferr                                        ; check for error
                54  00000609'EF    9E  0F5E   946             movab   directory,r4                    ; local storage for entry
                        0F65   947             getmem  (r6),(r4),#sdir$c_length                ; read into local storage
                        0F72   948             retiferr                                        ; check for error
                        0F76   949 5$:
                    1C  A4      DF  0F76   950             pushal  sdir$b_procinf(r4)              ; process information
                    1C  A4      95  0F79   951             tstb    sdir$b_procinf(r4)              ; check for ascii info
                        04      13  0F7C   952             beql    7$                              ; equal, don't display
                        10      DD  0F7E   953             pushl   #16
                        02      11  0F80   954             brb     8$                              ; branch around
                        00      DD  0F82   955 7$:         pushl   #0                              ; don't display info
                    0C  A4      DF  0F84   956 8$:         pushal  sdir$b_procnam(r4)              ; process name
                        10      DD  0F87   957             pushl   #16
                    2C  A4      DD  0F89   958             pushl   sdir$l_conid(r4)                ; connection id
                        56      DD  0F8C   959             pushl   r6                              ; address of the dir entry
                                    0F8E   960             print   4,<  !XL          !XL       !AD       !AD>
        00000000'EF         64      D1  0F98   961             cmpl    sdir$l_flink(r4),scs$gq_direct  ; any more entries
                        13      13  0FA2   962             beql    10$                             ; equal, no
                    56      64      D0  0FA4   963             movl    sdir$l_flink(r4),r6             ; get next entry address
                                    0FA7   964             getmem  (r6),(r4),#sdir$c_length        ; read into local storage
                    BF  50      E8  0FB4   965             blbs    r0,5$                           ; branch if read o.k.
                                    0FB7   966 ;
                                    0FB7   967 ; This is the second half of the summary page, a brief description of the system
                                    0FB7   968 ; blocks and the number of paths each system has.
```

```
                        0FB7   969 ;
                        0FB7   970 10$:
                        0FB7   971          skip    page
                        0FBE   972          skip    1
                        0FC7   973          print   0,<!_!_   --- SCS Systems Summary --->
                        0FD4   974          skip    1
                        0FDD   975          print   0,<  SB Address      Node       Type       System ID          Paths>
                        0FEA   976          print   0,<  ----------     ----       ----       ---------          ----->
                        0FF7   977          skip    1
                        1000   978
                        1000   979          getmem  @scs$gq_config,r7                          ; system block address
                        1010   980          retiferr
    58  00000645'EF   9E    1014   981          movab   sblock,r8                              ; local storage for sb
                        101B   982 15$:
                        101B   983          getmem  (r7),(r8),#sb$c_length                    ; read into local storage
                        102C   984          retiferr
                        1030   985
              39   10    1030   986          bsb     count_paths                            ; determine number of pb
              54   DD    1032   987          pushl   r4                                     ; move count onto stack
           18 A8   DD    1034   988          pushl   sb$b_systemid(r8)                      ; system id
        7E 1C A8   3C    1037   989          movzwl  sb$b_systemid+4(r8),-(sp)              ; high order 2 bytes
           24 A8   DF    103B   990          pushal  sb$t_swtype(r8)                        ; type of node
           24 A8   95    103E   991          tstb    sb$t_swtype(r8)                        ; check for missing type
              24   13    1041   992          beql    25$                                    ; equal, missing type
              04   DD    1043   993          pushl   #4                                     ;
           44 A8   9F    1045   994 16$:     pushab  sb$t_nodename(r8)                      ; node name
              57   DD    1048   995          pushl   r7                                     ; address of sb
                        104A   996          print   6,<  !XL       !6<!AC!>      !4<!AD!>       !XL!XW    !UL>
                        1057   997
           57   68   D0    1057   998          movl    sb$l_flink(r8),r7                      ; move to next sb
00000000'EF   57   D1    105A   999          cmpl    r7,scs$gq_config                       ; end of list of sb's
              B8   12    1061  1000          bneq    15$                                    ; branch if no
                        1063  1001 ;
                        1063  1002 ; We are done so let's get out of here.
                        1063  1003 ;
                        1063  1004 20$:
        50    01   D0    1063  1005          movl    #1,r0                                  ; return success
                   04    1066  1006          ret
                        1067  1007
                   00   DD    1067  1008 25$:     pushl   #0                          ; put zero length on stack for type
                   DA   11    1069  1009          brb     16$                         ; continue
                        106B  1010          .dsabl  lsb
                        106B  1011
                        106B  1012 Count_paths:
                   54   D4    106B  1013          clrl    r4                          ; counter for number of paths
     55    57   0C   C1    106D  1014          addl3   #sb$l_pbfl,r7,r5            ; start of list of pb's
        55     0C A8   D1    1071  1015          cmpl    sb$l_pbfl(r8),r5            ; any path blocks?
                   19   13    1075  1016          beql    35$                         ; equal, zero pb's
        56     0C A8   D0    1077  1017          movl    sb$l_pbfl(r8),r6            ; get next path block
                        107B  1018 25$:
                   54   96    107B  1019          incb    r4                          ; increment count
                        107D  1020          assume pb$l_flink eq 0
                        107D  1021          getmem  (r6)                        ; read in link to next pb
        55    51   D1    1086  1022          cmpl    r1,r5                       ; end of list? (assume statement for r1)
                   05   13    1089  1023          beql    35$                         ; equal, yes, so return to caller
        56    51   D0    108B  1024          movl    r1,r6                       ; follow the link (assume statement for r1)
                   EB   11    108E  1025          brb     25$                         ; still in loop of pb's
```

```
                1090  1026
 05             1090  1027 35$:    rsb                              ; return to main line code
```

```
                     1091  1029      .sbttl      display_sb_pbs --- display all system and path blocks
                     1091  1030 ;---
                     1091  1031 ;
                     1091  1032 ;       display_sb_pbs
                     1091  1033 ;
                     1091  1034 ;       This is a coroutine whose purpose is display each system
                     1091  1035 ;       block and its associated path blocks.
                     1091  1036 ;
                     1091  1037 ;       Inputs:
                     1091  1038 ;
                     1091  1039 ;               none
                     1091  1040 ;
                     1091  1041 ;       Outputs:
                     1091  1042 ;
                     1091  1043 ;               SCS data structures ( as mentioned above) are shown
                     1091  1044 ;               All registers are preserved.
                     1091  1045 ;
                     1091  1046 ;---
                     1091  1047 display_sb_pbs:
                0FFC 1091  1048         .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                     1093  1049
                     1093  1050         getmem  @scs$gq_config,r3                       ; system block address
                     10A3  1051         retiferr                                        ; return if error
                     10A7  1052 10$:
           53   DD   10A7  1053         pushl   r3                                      ; pass sb address
00000000'EF   01   FB   10A9  1054         calls   #1,show_system_block                    ; easy way out!
                     10B0  1055         assume  sb$l_flink eq 0
                     10B0  1056         getmem  (r3),r3                                 ; get next address
00000000'EF   53   D1   10BC  1057         cmpl    r3,scs$gq_config                        ; are we at the end?
              E2   12   10C3  1058         bneq    10$                                     ; not equal, still more
                     10C5  1059
        50   01   D0   10C5  1060         movl    #1,r0                                   ; return success
                  04   10C8  1061         ret
```

```
                                        .sbttl      show_connections --- display all connection descriptor tables (CDT)
                                ;---
                                ;
                                ;       show_connections
                                ;
                                ;       This is the main routine whose purpose is to display the contents
                                ;       of each connection descriptor table(CDT). A CDT is used to store
                                ;       information about a virtual circuit between two processes.  The
                                ;       first page is a brief summary of each cdt.
                                ;
                                ;       Inputs:
                                ;
                                ;               AP = pointer to TPARSE block
                                ;
                                ;       Outputs:
                                ;
                                ;               SCS data structures ( as mentioned above) are shown
                                ;               All registers are preserved.
                                ;
                                ;---
                                        .enabl lsb
                                null_string:
                             00         .byte   0
                                        .ascii //
                                show_connections::
                           0FFC         .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                ;
                                ;  Header information
                                ;
                                        subhd   <VAXcluster data structures>            ; set heading
                                        skip    page
                                        skip    1
                                        print   0,<!_!_  --- CDT Summary Page --->
                                        skip    1
                                        print   0,<CDT Address   Local Process     Connection ID     State       Rem
                                        print   0,<-----------   -------------     -------------     -----       ---
                                        skip    1
                                ;
                                ;  Now set up the data structures.  Read the cdl ( specifically the location
                                ;  containing the first free cdt and the list of cdts).  Then read into local
                                ;  storage the first cdt to display.  Check to see if this cdt is on the free
                                ;  list, if it is then it will not be displayed.  Also if the state of the cdt
                                ;  is closed, it will not be displayed.  Otherwise it will be displayed.
                                ;
                                        getmem  @scs$gl_cdl,r6          ; address of cdl
                 00000004'EF    D4      clrl    cdl_size                ; initialize field
                                        getmem  cdl$w_size(r6),cdl_size,#2 ; size of cdl to read into virtual memory
                                        retiferr                        ; return on error
                 00000000'EF    9F      pushab  cdl                     ; will contain virtual address for cdl
                 00000004'EF    9F      pushab  cdl_size                ; size of cdl
              00000000'GF   02  FB      calls   #2,g^lib$get_vm         ; get memory for cdl
                                        retiferr                        ; return on error
                                        getmem  -16(r6),@cdl,cdl_size   ; read cdl into storage
                                        retiferr                        ; return on error
 56  00000000'EF   00000004'EF  C1      addl3   cdl_size,cdl,r6         ; r6 => end address of cdl
     55  00000000'EF   10       C1      addl3   #16,cdl,r5              ; base of cdl
```

```
                5A   FD A5    D0  1193  1120          movl    cdl$w_maxconidx(r5),r10 ; max number of cdts in table
                          3B    19  1197  1121          blss    20$                     ; no cdts in table
                     57   55    D0  1199  1122          movl    r5,r7                   ; save address of cdl
                                    119C  1123
                          5B    D4  119C  1124          clrl    r11                     ; counter of the number of free cdt's
                     56   57    D1  119E  1125 5$:      cmpl    r7,r6                   ; safety check for end of list
                          31    13  11A1  1126          beql    20$                     ; yes, exit loop
                        00B2    30  11A3  1127          bsbw    free_cdt_list           ; check to see if on free cdt list
                      05 50     E8  11A6  1128          blbs    r0,10$                  ; set, not on free list so display
                      F2 5A     F4  11A9  1129          sobgeq  r10,5$                  ; on free list so don't display
                          26    11  11AC  1130          brb     20$                     ; hit end of cdts
59   00000008'EF     9E  11AE  1131 10$:     movab   cdt,r9                  ; local storage for cdt
                                    11B5  1132          getmem  @(r7)+,(r9),#cdt$c_length ; read into local storage
                          5A    DD  11C6  1133          pushl   r10                     ; save r10
                00001281'EF     16  11C8  1134          jsb     display_sumline         ; display one line of the summary page
                     5A     8E    D0  11CE  1135          movl    (sp)+,r10               ; restore r10
                        CA 5A     F4  11D1  1136          sobgeq  r10,5$                  ; get next cdt
                                    11D4  1137 ;
                                    11D4  1138 ;  At this stage the summary page is almost complete.  The last thing to display
                                    11D4  1139 ;  is the number of free cdt's.  R11 contains this number.
                                    11D4  1140 ;
                                    11D4  1141 20$:     skip    1
                          5B    DD  11DD  1142          pushl   r11
                                    11DF  1143          print   1,<Number of free CDT's:  !UL>
                                    11EC  1144 ;
                                    11EC  1145 ;  The summary page is done.  Now we will again loop through all the
                                    11EC  1146 ;  cdts in use and call upon a routine to display the full contents
                                    11EC  1147 ;  of each and every one of them.
                                    11EC  1148 ;
                54   F4 A5    D0  11EC  1149          movl    cdl$l_freecdt(r5),r4    ; first free cdt
                5A   F0 A5    D0  11F0  1150          movl    cdl$w_maxconidx(r5),r10 ; max number of cdts in table
                     56   55    D1  11F4  1151 30$:     cmpl    r5,r6                   ; test for end of list
                          48    13  11F7  1152          beql    40$                     ; equal, end of list
                     54   65    D1  11F9  1153          cmpl    (r5),r4                 ; beginning of free cdt's
                          2D    13  11FC  1154          beql    35$                     ; equal, update r6 to next free cdt
                                    11FE  1155          skip    page                    ; new page for each cdt
                                    1205  1156 ;
                                    1205  1157 ;  First check the state to see if this cdt is marked as closed.  If it is
                                    1205  1158 ;  don't bother to display.
                                    1205  1159 ;
            57   65   28    C1  1205  1160          addl3   #cdt$w_state,(r5),r7    ; point to cdt state
                                    1209  1161          getmem  (r7),r7                 ; read in state
                                    1215  1162          retiferr                        ; return on error
                     00   57    B1  1219  1163          cmpw    r7,#cdt$c_closed        ; closed cdt?
                          1E    13  121C  1164          beql    38$                     ; equal, disregard closed cdt
                                    121E  1165
                     7E   85    D0  121E  1166          movl    (r5)+,-(sp)             ; address of cdt
          000013A9'EF     01    FB  1221  1167          calls   #1,display_cdt          ; display
                        C9 5A     F4  1228  1168          sobgeq  r10,30$                 ; get next one
                                    122B  1169 35$:     getmem  @(r5)+,r4               ; get next free cdt in r6
                        BA 5A     F4  1237  1170          sobgeq  r10,30$                 ; loop for next cdt
                          05    11  123A  1171          brb     40$                     ; end of list
                                    123C  1172
                          85    D5  123C  1173 38$:     tstl    (r5)+                   ; increment r5
                        B3 5A     F4  123E  1174          sobgeq  r10,30$                 ; loop for next valid cdt
                                    1241  1175
                                    1241  1176 40$:
```

```
                                1241  1177                pushab  cdl                             ; address of virtual memory to deall.
                                1247  1178                pushab  cdl_size
                        02      124D  1179                calls   #2,g^lib$free_vm                ; deallocate virtual memory
                    50  01      1254  1180                movl    #1,r0                           ; success
                        04      1257  1181                ret                                     ; finished!!!
                                1258  1182                .dsabl  lsb
                                1258  1183
                                1258  1184
                                1258  1185 free_cdt_list:
                    50  01      1258  1186                movl    #1,r0                           ; assume not on the cdt free list
                54  F4  A5      125B  1187                movl    cdl[$l_freecdt(r5),r4           ; head of list
                    54  67      125F  1188 5$:            cmpl    (r7),r4                         ; on the free list
                        16      1262  1189                beql    10$                             ; equal, on free list
                                1264  1190                getmem  (r4),r4                         ; chain down the list
                                1270  1191                retiferr                                ; return on error
                            54  1274  1192                tstl    r4                              ; end of list
                            08  1276  1193                beql    15$                             ; yes end of list and no match
                            E5  1278  1194                brb     5$                              ; loop and compare
                            87  127A  1195 10$:           tstl    (r7)+                           ; point to next cdt in list
                            5B  127C  1196                incl    r11                             ; increment the counter of free cdts
                            50  127E  1197                clrl    r0                              ; r0 clr indicates free list
                                1280  1198 15$:           rsb                                     ; return to caller
                                1281  1199
```

```
                        1281  1201      .sbttl      display_sumline --- display a line of the cdt summary page
                        1281  1202 ;---
                        1281  1203 ;
                        1281  1204 ;       display_sumline
                        1281  1205 ;
                        1281  1206 ;       This is the subroutine whose purpose is to display a line of the
                        1281  1207 ;       summary page for the given cdt.
                        1281  1208 ;
                        1281  1209 ;       Inputs:
                        1281  1210 ;
                        1281  1211 ;               R9 = address of cdt in local storage
                        1281  1212 ;
                        1281  1213 ;       Outputs:
                        1281  1214 ;
                        1281  1215 ;               A line of the cdt summary page is displayed.
                        1281  1216 ;
                        1281  1217 ;---
                        1281  1218         .enabl  lsb
                        1281  1219 display_sumline:
00   28 A9   B1         1281  1220         cmpw    cdt$w_state(r9),#cdt$c_closed   ; if closed, ignore this cdt
          03   12       1285  1221         bneq    1$                              ; if not equal, cdt is of interest
        003F   31       1287  1222         brw     40$                             ; neq, not closed so continue to process
                        128A  1223 ;
                        128A  1224 ;  Now obtain the remote node name.  This requires going through a few
                        128A  1225 ;  channels.  The CDT contains the path block address.  The path block
                        128A  1226 ;  will lead us to the system block which yields the remote node name.
                        128A  1227 ;  Follow me.  However if the cdt is a listen cdt,  the remote node name
                        128A  1228 ;  will not be present.  So test first for this condition.
                        128A  1229 ;
01   28 A9   B1         128A  1230 1$:     cmpw    cdt$w_state(r9),#cdt$c_listen           ; listen cdt?
          06   12       128E  1231         bneq    5$                              ; no equal, not a listen cdt
      FE35 CF   9F       1290  1232         pushab  null_string                     ; listen cdt - no remote node name
           0B   11       1294  1233         brb     10$                             ;
                        1296  1234 5$:
           59   DD       1296  1235         pushl   r9                              ; address of cdt in local storage
0000136D'EF     01   FB  1298  1236         calls   #1,remote_node                  ; find remote node name
           5A   DD       129F  1237         pushl   r10                             ; address of counted ascii string
                        12A1  1238 ;
                        12A1  1239 ;  Cdt's are not the easiest data structure to display.  The key to
                        12A1  1240 ;  displaying them correctly is the state and blkstate fields.  The
                        12A1  1241 ;  strategy is to translate the blkstate field first.  If this succeeds,
                        12A1  1242 ;  then displaying the cdt follows the normal path.  If this translation
                        12A1  1243 ;  fails,  then the state field is translated.  The blkst_flag will be
                        12A1  1244 ;  clear if the translation of it succeeded.  Setting of the flag indicates
                        12A1  1245 ;  failure.
                        12A1  1246 ;
000012CA'EF     16      12A1  1247 10$:    jsb     state_translate                 ; translate state value
           50   DD       12A7  1248         pushl   r0                              ; address of counted ascii string
        18 A9   DD       12A9  1249         pushl   cdt$l_lconid(r9)                ; connection id
                        12AC  1250 ;
                        12AC  1251 ;  Now to display the local process name.  This also is not as straight forward
                        12AC  1252 ;  as it seems.  If the cdt is NOT a listen CDT, then the process name in the
                        12AC  1253 ;  cdt is valid.  However if the cdt is a listen cdt, the local process name
                        12AC  1254 ;  field will most likely be zero. (Bummer!)  There is hope however for re-
                        12AC  1255 ;  trieving the name.  The cdt will have a valid connection id.  Therefore,
                        12AC  1256 ;  if search the scs directory looking for a match on connection id,  the
                        12AC  1257 ;  entry which matches based on the id will contain the local process name.
```

```
                        12AC  1258 ;
                59   DD  12AC  1259          pushl   r9                              ; address of cdt in local storage
      000012EB'EF   01   FB  12AE  1260          calls   #1,find_procname                ; find the local process name
                52   DD  12B5  1261          pushl   r2                              ; address of process name
                53   DD  12B7  1262          pushl   r3                              ; maximum length of name
              FC A7  DD  12B9  1263          pushl   -4(r7)                          ; address of cdt
                         12BC  1264          print   6,< !XL     !AD     !XL         !9<!AC!>    !AC>
                         12C9  1265 40$:     rsb                                     ; return to main routine
                         12CA  1266          .dsabl  lsb
```

```
                     12CA  1268      .sbttl      state_translate --- translate cdt state values to names
                     12CA  1269 ;---
                     12CA  1270 ;
                     12CA  1271 ;       state_translate
                     12CA  1272 ;
                     12CA  1273 ;       This is the subroutine whose purpose is to translate the state fields
                     12CA  1274 ;       in the given cdt to their corresponding ascii names for display.
                     12CA  1275 ;
                     12CA  1276 ;       Inputs:
                     12CA  1277 ;
                     12CA  1278 ;              R9 = address of cdt in local storage
                     12CA  1279 ;
                     12CA  1280 ;       Outputs:
                     12CA  1281 ;
                     12CA  1282 ;              State fields in the cdt are translated to meaningful
                     12CA  1283 ;              ascii names.
                     12CA  1284 ;
                     12CA  1285 ;---
                     12CA  1286 state_translate:
                     12CA  1287 ;
                     12CA  1288 ;  Translate the SCS blocked send state location to their equivalent
                     12CA  1289 ;  ascii names.
                     12CA  1290 ;
   52    2A A9   3C  12CA  1291          movzwl  cdt$w_blkstate(r9),r2   ; scs send blocked state
53    EFF6 CF   9E  12CE  1292          movab   cdt_blkstate,r3         ; definition table
 00000000'GF   16  12D3  1293          jsb     g^translate_address     ; translate constants to names
          0F   12  12D9  1294          bneq    10$                     ; not equal, match found
                     12DB  1295 ;
                     12DB  1296 ;  The translation has failed.  Translate the state value and set the flag
                     12DB  1297 ;  to indicate that the blkstate translation has failed.
                     12DB  1298 ;
   52    28 A9   3C  12DB  1299          movzwl  cdt$w_state(r9),r2      ; state value
53    EF75 CF   9E  12DF  1300          movab   cdt_state,r3            ; corresponding definition table
 00000000'GF   16  12E4  1301          jsb     g^translate_address     ; translate
                     12EA  1302 10$:
                05   12EA  1303          rsb
```

```
                        12EB  1305      .sbttl      find_procname --- Find the local process name.
                        12EB  1306 ;---
                        12EB  1307 ;
                        12EB  1308 ;       find_procname
                        12EB  1309 ;
                        12EB  1310 ;       This is the coroutine whose purpose is to find the local process
                        12EB  1311 ;       name.  If the cdt is a listen cdt, then the local process name in
                        12EB  1312 ;       the cdt will not be valid.  We will have to use the scs directory
                        12EB  1313 ;       to determine the name.  The connection id of the cdt is used to find
                        12EB  1314 ;       the scs directory entry with the local process name for this cdt.
                        12EB  1315 ;       If the state of the cdt is other than listen,  then the field in
                        12EB  1316 ;       the cdt should be valid.
                        12EB  1317 ;
                        12EB  1318 ;       Inputs:
                        12EB  1319 ;
                        12EB  1320 ;               4(AP) = address of cdt in local storage
                        12EB  1321 ;
                        12EB  1322 ;       Outputs:
                        12EB  1323 ;
                        12EB  1324 ;               R2 = address of the local process name.
                        12EB  1325 ;               R3 = length of the name.
                        12EB  1326 ;               All other registers are preserved.
                        12EB  1327 ;
                        12EB  1328 ;---
                        12EB  1329 find_procname::
                0030    12EB  1330          .word   ^m<r4,r5>
                        12ED  1331
          55    04 AC   D0    12ED  1332          movl    4(ap),r5                    ; get the address of the cdt
          01    28 A5   B1    12F1  1333          cmpw    cdt$w_state(r5),#cdt$c_listen   ; is this a listen cdt
                   2B   13    12F5  1334          beql    15$                         ; equal, yes
                54 A5   D5    12F7  1335          tstl    cdt$l_lprocnam(r5)          ; test for zero address
                   1B   13    12FA  1336          beql    10$                         ; no local process name
    52    00000685'EF   9E    12FC  1337          movab   procname,r2                 ; address of local process
                        1303  1338          getmem  @cdt$l_lprocnam(r5),(r2),#16    ; read into local storage
             53   10    D0    1311  1339          movl    #16,r3
                0055    31    1314  1340          brw     50$
          52  FDAE CF   9E    1317  1341 10$:     movab   null_string,r2              ; no local process name
             53   00    D0    131C  1342          movl    #0,r3                       ; zero length
                004A    31    131F  1343          brw     50$
                        1322  1344 ;
                        1322  1345 ;  Search the scs directory for the entry with the same connection id.
                        1322  1346 ;
    54  00000609'EF   9E    1322  1347 15$:     movab   directory,r4                ; local storage for directory entry
                        1329  1348          getmem  @scs$gq_direct,(r4),#sdir$c_length
                        133A  1349          retiferr                            ; return if error
    2C A4    18 A5   D1    133E  1350 20$:     cmpl    cdt$l_lconid(r5),sdir$l_conid(r4)
                   16   13    1343  1351          beql    30$                         ; match
             53   64    D0    1345  1352          movl    sdir$l_flink(r4),r3         ; next entry
                        1348  1353          getmem  (r3),(r4),#sdir$c_length    ; read into local storage
                        1355  1354          retiferr                            ; return on error
                   E3   11    1359  1355          brb     20$
                        135B  1356 ;
                0C A4   95    135B  1357 30$:     tstb    sdir$b_procnam(r4)          ; check if name exists
                   05   12    135E  1358          bneq    40$                         ; not equal, exists
             53   00    D0    1360  1359          movl    #0,r3                       ; zero length
                   03   11    1363  1360          brb     45$                         ; branch around
             53   10    D0    1365  1361 40$:     movl    #16,r3                      ; length
```

```
                52   0C A4   9E  1368  1362 45$:    movab   sdir$b_procnam(r4),r2   ; address of local process name
                             04  136C  1363 50$:    ret
```

```
                        136D  1365      .sbttl      remote_node --- find the remote node name
                        136D  1366 ;---
                        136D  1367 ;
                        136D  1368 ;       remote_node
                        136D  1369 ;
                        136D  1370 ;               This routine is to determine the name of the remote node
                        136D  1371 ;               given the contents of a cdt.  The cdt will give us the path
                        136D  1372 ;               block address which in turn will yield the system block
                        136D  1373 ;               address.  The system block contains the remote node name
                        136D  1374 ;               that is desired.
                        136D  1375 ;
                        136D  1376 ;       Inputs:
                        136D  1377 ;
                        136D  1378 ;               4(ap) = address of cdt in local storage
                        136D  1379 ;
                        136D  1380 ;       Outputs:
                        136D  1381 ;
                        136D  1382 ;               R10 = address of counted ascii string of remote node name.
                        136D  1383 ;               All other registers are preserved.
                        136D  1384 ;
                        136D  1385 ;---
                        136D  1386 remote_node:
                  033C  136D  1387           .word   ^m<r2,r3,r4,r5,r8,r9>
          59    04 AC   D0  136F  1388           movl    4(ap),r9                ; address of cdt in local storage
          58    1C A9   D0  1373  1389           movl    cdt$l_pb(r9),r8         ; path block address
             58    30   C0  1377  1390           addl2   #pb$l_sblink,r8         ; point to system block address
                        137A  1391           getmem  (r8),r8                 ; yields system block address
                        1386  1392           retiferr                        ; return if error
    5A  000006A5'EF     9E  138A  1393           movab   node,r10                ; local storage for node name
    58  00000044 8F     C0  1391  1394           addl2   #sb$t_nodename,r8       ; point at node name
                        1398  1395           getmem  (r8),(r10),#sb$s_nodename ; read it into local storage
             50   01    D0  13A5  1396           movl    #1,r0                   ; success
                        04  13A8  1397           ret                             ; return with address of node name in r10
                            13A9  1398
```

```
                                                M 1
                     13A9  1400     .sbttl      display_cdt --- display a connection descriptor table
                     13A9  1401 ;---
                     13A9  1402 ;
                     13A9  1403 ;       display_cdt
                     13A9  1404 ;
                     13A9  1405 ;       This is a coroutine whose purpose is display each connection
                     13A9  1406 ;       descriptor table (CDT). A CDT is used to store information
                     13A9  1407 ;       about a virtual circuit between two processes.
                     13A9  1408 ;
                     13A9  1409 ;       Inputs:
                     13A9  1410 ;
                     13A9  1411 ;               4(ap) = actual address of cdt
                     13A9  1412 ;
                     13A9  1413 ;       Outputs:
                     13A9  1414 ;
                     13A9  1415 ;               CDT is displayed.
                     13A9  1416 ;               All registers are preserved.
                     13A9  1417 ;
                     13A9  1418 ;---
                     13A9  1419         .enabl  lsb
                     13A9  1420 display_cdt::
                OFFC 13A9  1421         .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                     13AB  1422
     59 00000008'EF   9E 13AB  1423         movab   cdt,r9                          ; local storage for cdt
                     13B2  1424         getmem  @4(ap),(r9),#cdt$c_length       ; read into local storage
                     13C4  1425         retiferr
                     13C8  1426         skip    1
           04 AC     DD 13D1  1427         pushl   4(ap)           ; actual address of cdt
                     13D4  1428         print   0,<!_  --- Connection Descriptor Table (CDT) !XL --->
                     13E1  1429         skip    1
                     13EA  1430 ;
                     13EA  1431 ;  Determine local process name
                     13EA  1432 ;
      00   28 A9     B1 13EA  1433         cmpw    cdt$w_state(r9),#cdt$c_closed   ; if closed, no local process name
                  0A 12 13EE  1434         bneq    1$                              ; not equal, continue to look
      00000685'EF   9F 13F0  1435         pushab  procname                        ; null process name
                  00 DD 13F6  1436         pushl   #0                              ; zero length
                  21 11 13F8  1437         brb     3$                              ; no local process name available
            54 A9    D5 13FA  1438 1$:     tstl    cdt$l_lprocnam(r9)              ; check to see if valid address
                  03 12 13FD  1439         bneq    2$                              ; yes there is a process name
                011E 31 13FF  1440         brw     40$                             ; go and search for it
                     1402  1441
     52 00000685'EF   9E 1402  1442 2$:     movab   procname,r2                     ; put into local storage
                     1409  1443         getmem  @cdt$l_lprocnam(r9),(r2),#16    ; local process name
                  52 DD 1417  1444         pushl   r2                      ; push address of process name
                  10 DD 1419  1445         pushl   #16                             ; length of name
                     141B  1446 ;
                     141B  1447 ;  Translate state value to ascii string
                     141B  1448 ;
      52   28 A9     3C 141B  1449 3$:     movzwl  cdt$w_state(r9),r2      ; state value
     53   EE35 CF   9E 141F  1450         movab   cdt_state,r3            ; corresponding definition table
      00000000'GF   16 1424  1451         jsb     g^translate_address     ; translate
                  03 12 142A  1452         bneq    4$                      ; match found
                00EA 31 142C  1453         brw     35$                     ; no match
                  50 DD 142F  1454 4$:     pushl   r0                      ; translated to names
        7E   28 A9     3C 1431  1455 5$:     movzwl  cdt$w_state(r9),-(sp)   ; state value
                     1435  1456         print   3,<State:  !XW !AC                      Local Process:        !AD>
```

```
                                1442  1457                                                    ; display
                                1442  1458 ;
                                1442  1459 ;  Determine remote process name if it exists.
                                1442  1460 ;
          00    28 A9    B1     1442  1461          cmpw    cdt$w_state(r9),#cdt$c_closed   ; check for closed cdt
                   0A    12     1446  1462          bneq    10$                             ; not equal, continue to look for
          000006B5'EF    9F     1448  1463          pushab  procname                        ; null process name
                   00    DD     144E  1464          pushl   #0                              ; zero length
                   27    11     1450  1465          brb     15$                             ; no remote node and process available
                50 A9    D5     1452  1466 10$:     tstl    cdt$l_rprocnam(r9)              ; check for non-zero address
                   22    13     1455  1467          beql    15$                             ; equal, remote process name not available
       52   000006B5'EF    9E     1457  1468          movab   procname,r2                     ; local storage for remote process name
                         145E  1469          getmem  @cdt$l_rprocnam(r9),(r2),#16    ; read into local storage
                   52    DD     146C  1470          pushl   r2                              ; address of remote process name
                   10    DD     146E  1471          pushl   #16                             ; length of name
                         1470  1472 ;
                         1470  1473 ;  Obtain the remote node name.
                         1470  1474 ;
                   59    DD     1470  1475          pushl   r9                              ; address of cdt in local storage
             FEF6 CF    01    FB     1472  1476          calls   #1,remote_node                  ; find remote node
                   5A    DD     1477  1477          pushl   r10                             ; counted ascii string
                         1479  1478 ;
                         1479  1479 ;  Translate scs blocked send state to ascii string
                         1479  1480 ;
             52    2A A9    3C     1479  1481 15$:     movzwl  cdt$w_blkstate(r9),r2   ; scs send blocked state
          53   EE47 CF    9E     147D  1482          movab   cdt_blkstate,r3         ; definition table
           00000000'GF    16     1482  1483          jsb     g^translate_address     ; translate constants to names
                   03    12     1488  1484          bneq    19$                             ; translate failed if equal
                0085    31     148A  1485          brw     30$
                   50    DD     148D  1486 19$:     pushl   r0                              ; address of counted ascii string
       7E    2A A9    3C     148F  1487 20$:     movzwl  cdt$w_blkstate(r9),-(sp)        ; scs send blocked state
                         1493  1488 ;
                         1493  1489 ;  Display
                         1493  1490 ;
       01    28 A9    B1     1493  1491          cmpw    cdt$w_state(r9),#cdt$c_listen   ; check for listen cdt
                   15    13     1497  1492          beql    22$                             ; equal, listen so no remote
       00    28 A9    B1     1499  1493          cmpw    cdt$w_state(r9),#cdt$c_closed   ; check for closed cdt
                   0F    13     149D  1494          beql    22$                             ; no remote for closed
                         149F  1495          print   5,<Blocked State:  !XW !AC              Remote Node::Process: !AC::!A
                   0D    11     14AC  1496          brb     23$
                         14AE  1497 22$:     print   2,<Blocked State:  !XW !AC>
                         14BB  1498
                         14BB  1499 23$:     make_symbol     PB, cdt$l_pb(r9)
                         14D1  1500          make_symbol     PDT, cdt$l_pdt(r9)
                         14E7  1501          skip    1
                         14F0  1502
                         14F0  1503          print_columns -
                         14F0  1504                  (r9),4(ap),-
                         14F0  1505                  cdt_col_1,cdt_col_2,cdt_col_3           ; display cdt
          50    01    D0     150E  1506 25$:     movl    #1,r0                           ; return with success
                   04    1511  1507          ret                                     ; done
                         1512  1508
          FBB3 CF    9F     1512  1509 30$:     pushab  null_string
             FF76    31     1516  1510          brw     20$
                         1519  1511
          FBAC CF    9F     1519  1512 35$:     pushab  null_string             ; translation failed
             FF11    31     151D  1513          brw     5$                              ; return to main line code
```

```
                    1520  1514
              59    DD  1520  1515 40$:    pushl   r9                      ; local address of cdt
    FDC4 CF   01    FB  1522  1516         calls   #1,find_procname         ; maybe in the directory entry
              52    DD  1527  1517         pushl   r2                      ; address of local process name
              53    DD  1529  1518         pushl   r3                      ; length of name
            FEED    31  152B  1519         brw     3$                      ; return to main line code
                        152E  1520
                        152E  1521         .dsabl  lsb
```

```
                     152E  1523     .sbttl      cdt_byaddr --- display the cdt requested by the user
                     152E  1524 ;---
                     152E  1525 ;
                     152E  1526 ;       cdt_byaddr
                     152E  1527 ;
                     152E  1528 ;       This is a routine whose purpose is display a connection
                     152E  1529 ;       descriptor table (CDT) which the user has requested by using
                     152E  1530 ;       the /ADDR qualifier and a valid address of a cdt. A CDT is
                     152E  1531 ;       used to store information about a virtual circuit between
                     152E  1532 ;       two processes.
                     152E  1533 ;
                     152E  1534 ;       Inputs:
                     152E  1535 ;
                     152E  1536 ;               AP = TPARSE block (TPA$L_NUMBER contains the address)
                     152E  1537 ;
                     152E  1538 ;       Outputs:
                     152E  1539 ;
                     152E  1540 ;               The requested CDT is displayed if a valid address is specified.
                     152E  1541 ;               Otherwise an informational message is sent to say invalid cdt
                     152E  1542 ;               address.
                     152E  1543 ;               All registers are preserved.
                     152E  1544 ;
                     152E  1545 ;---
                     152E  1546          .enabl  lsb
                     152E  1547 cdt_byaddr::
                0FFC 152E  1548          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
          57    1C AC  D0  1530  1549          movl    tpa$l_number(ap),r7             ; get address of cdt
0000154F'EF     16       1534  1550          jsb     verify_cdt                      ; is this a cdt
          0E 50  E9       153A  1551          blbc    r0,10$                          ; clear, not a cdt
                     153D  1552 ;
                     153D  1553 ;  Now that we are through the validation phase the rest is trivial.
                     153D  1554 ;
                     153D  1555          skip    page
          57    DD       1544  1556          pushl   r7                              ; pass the actual address
    FE5E CF     01   FB  1546  1557          calls   #1, display_cdt                 ; display it
                     154B  1558 10$:
          50    01   D0  154B  1559          movl    #1,r0
                     04  154E  1560          ret
                     154F  1561
                     154F  1562 verify_cdt:
                     57   D5  154F  1563          tstl    r7                              ; check for 80000000 address
                     28   18  1551  1564          bgeq    900$                            ; not valid
          58   57    0A   C1  1553  1565          addl3   #cdt$b_type,r7,r8               ; point at the type field
                     1557  1566          getmem  (r8),r8                         ; attempt to read
                     1563  1567          retiferr                                ; return on error
        60 8F    58   91  1567  1568          cmpb    r8,#dyn$c_scs                   ; check for the right type
                     0E   12  156B  1569          bneq    900$                            ; not equal, error
    58  58   F8 8F   78  156D  1570          ashl    #-8,r8,r8                       ; point at subtype
              02     58   91  1572  1571          cmpb    r8,#dyn$c_scs_cdt               ; check for correct subtype
                     04   12  1575  1572          bneq    900$                            ; not equal, invalid address
          50    01   D0  1577  1573          movl    #1,r0                           ; valid cdt
                     05  157A  1574          rsb
                     157B  1575
                     57   DD  157B  1576 900$:   pushl   r7                              ; invalid cdt address
                     157D  1577          type    1,<!XL is not the address of a CDT>
          50    00   D0  15C5  1578          movl    #0,r0                           ; invalid cdt
                     05  15C8  1579          rsb
```

```
                15C9  1580
                15C9  1581          .dsabl  lsb
```

```
                        1583      .sbttl      connection descriptor tables & action routines
                        1584 ;
                        1585 ;  PRINT_COLUMNS table for CDT displays
                        1586 ;
                        1587
                        1588
                        1589 cdt_col_1:
                        1590          column_list     -
                        1591                  cdt$, 16, 8, 4, < -
                        1592                  <<Local Con. ID>,l_lconid,xl>,-
                        1593                  <<Remote Con. ID>,l_rconid,xl>,-
                        1594                  <<Receive Credit>,w_rec,uw>,-
                        1595                  <<Send Credit>,w_send,uw>,-
                        1596                  <<Min. Rec. Credit>,w_minrec,uw>,-
                        1597                  <<Pend Rec. Credit>,w_pendrec,uw>,-
                        1598                  <<Initial Rec. Credit>,w_initlrec,uw,20,4>,-
                        1599                  <<Rem. Sta.>,cdt_6bytes,cdt$b_rstation,12,12>, -
                        1600                  <<Rej/Disconn Reason>,w_reason,uw,20,4>,-
                        1601                  <<Queued for BDT>,w_qbdt_cnt,uw>,-
                        1602                  <<Queued Send Credit>,w_qcr_cnt,uw,20,4>,-
                        1603                  >
                        1604
                        1605 cdt_col_2:
                        1606          column_list     -
                        1607                  cdt$, 16, 8, 4, < -
                        1608                  <<Datagrams sent>,l_dgsent,ul>,-
                        1609                  <<Datagrams rcvd>,l_dgrcvd,ul>,-
                        1610                  <<Datagram discard>,l_dgdiscard,ul>,-
                        1611                  <<Messages Sent>,l_msgsent,ul>,-
                        1612                  <<Messages Rcvd.>,l_msgrcvd,ul>,-
                        1613                  <<Send Data Init.>,l_snddats,ul>,-
                        1614                  <<Req Data Init.>,l_reqdats,ul>,-
                        1615                  <<Bytes Sent>,l_bytsent,ul>,-
                        1616                  <<Bytes rcvd>,l_bytreqd,ul>,-
                        1617                  <<Total bytes map>,l_bytmapd,ul>,-
                        1618                  >
                        1619
                        1620 cdt_col_3:
                        1621          column_list     -
                        1622                  cdt$, 16, 8, 0, < -
                        1623                  <<Message queue>,l_waitqfl,q2>,-
                        1624                  <<Send Credit Q.>,l_crwaitqfl,q2>,-
                        1625                  <<PB address>,l_pb,xl>,-
                        1626                  <<PDT address>,l_pdt,xl>,-
                        1627                  <<Error Notify>,l_erraddr,xl>,-
                        1628                  <<Receive Buffer>,l_scsmsg,xl>,-
                        1629                  <<Connect Data>, l_condat,xl>,-
                        1630                  <<Aux. Structure>,l_auxstruc,xl>,-
                        1631                  >
                        1632 ;
                        1633 ;  The following are all PRINT_COLUMNS action routines for the show
                        1634 ;  connection descriptor table displays.
                        1635 ;
                        1636 ;  Action Routine Inputs:
                        1637 ;
                        1638 ;       R2      value from the COLUMN_LIST entry
                        1639 ;       R5      size of value section for this item
```

```
                     17C9  1640 ;       R7      address of a descriptor for a scratch string in
                     17C9  1641 ;               which the FAO converted value is to be returned
                     17C9  1642 ;       R11     base address of the local CDT copy
                     17C9  1643 ;
                     17C9  1644 ;   Action Routine Outputs:
                     17C9  1645 ;
                     17C9  1646 ;       R0      status
                     17C9  1647 ;               lbs ==> use this entry
                     17C9  1648 ;               lbc ==> skip this entry
                     17C9  1649 ;       R1-R5   scratch
                     17C9  1650 ;               all other registers must be preserved
                     17C9  1651 ;
                     17C9  1652 cdt_fao_6bytes:
                     17C9  1653         string  <!XW!XL>
                     17D7  1654
                     17D7  1655 ;*********
                     17D7  1656 cdt_6bytes:
53   5B   52   C1   17D7  1657         addl3   r2,r11,r3               ; locate storage of interest
     55   0C   C2   17DB  1658         subl    #12,r5                  ; get size of filler field
                     17DE  1659         $fao_s  -
                     17DE  1660                 ctrstr = cdt_fao_6bytes,-
                     17DE  1661                 outbuf = (r7),-
                     17DE  1662                 outlen = (r7),-
                     17DE  1663                 p1 = r5,-
                     17DE  1664                 p2 = (r3),-
                     17DE  1665                 p3 = 4(r3)
                05   17F3  1666         rsb
```

```
                        17F4  1668      .sbttl      show_rspid --- display RDT entries
                        17F4  1669 ;---
                        17F4  1670 ;
                        17F4  1671 ;       show_rspid
                        17F4  1672 ;
                        17F4  1673 ;       This is the main routine whose purpose is to display the contents
                        17F4  1674 ;       of the response descriptor table(RDT).  If an address of a connection
                        17F4  1675 ;       descriptor table is specified, then only those entries with the same
                        17F4  1676 ;       cdt will be displayed.
                        17F4  1677 ;
                        17F4  1678 ;
                        17F4  1679 ;       Inputs:
                        17F4  1680 ;
                        17F4  1681 ;               AP = pointer to TPARSE block
                        17F4  1682 ;               CDT_SPCFY = 1 means /CONNECTION qualifier was present
                        17F4  1683 ;                           0 means the qualifier was absent
                        17F4  1684 ;
                        17F4  1685 ;       Outputs:
                        17F4  1686 ;
                        17F4  1687 ;               All RDT entries are displayed unless they are on the free list
                        17F4  1688 ;               or are permanently allocated in which cases they are of no
                        17F4  1689 ;               interest to us.
                        17F4  1690 ;               All registers are preserved.
                        17F4  1691 ;
                        17F4  1692 ;---
                        17F4  1693          .enabl lsb
                        17F4  1694
                        17F4  1695 show_rspid::
                0FFC    17F4  1696          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        17F6  1697
              57    D4  17F6  1698          clrl    r7                              ; initialize cdt address
000006F9'     EF    D5  17F8  1699          tstl    cdt_spcfy                       ; cdt address specified?
                14    13  17FE  1700          beql    5$                              ; equal, no address
          57  1C AC    D0  1800  1701          movl    tpa$l_number(ap),r7             ; cdt address
000006F9'     EF    D4  1804  1702          clrl    cdt_spcfy                       ; clear for next round
            FD41  CF    16  180A  1703          jsb     verify_cdt                      ; check validity of cdt
              03  50    E8  180E  1704          blbs    r0,5$                           ; valid cdt
                010D    31  1811  1705          brw     40$                             ; not a cdt
                        1814  1706 ;
                        1814  1707 ;  Header information
                        1814  1708 ;
                        1814  1709 5$:     subhd   <VAXcluster data structures>            ; set heading
                        1821  1710          skip    page
                        1828  1711          skip    1
                        1831  1712          getmem  @scs$gl_rdt,r6          ; address of rdt
                        1841  1713          retiferr
              56    DD  1845  1714          pushl   r6
                        1847  1715          print   1,<!_--- Summary of Response Descriptor Table(RDT) !XL --->
                        1854  1716          skip    1
                        185D  1717          print   0,<RSPID      CDRP Address     CDT Address     Local Process Name
                        186A  1718          print   0,<-----      ------------     -----------     ------------------
                        1877  1719          skip    1
                        1880  1720 ;
                        1880  1721 ;  Now set up the data structures.  Read the rdt ( specifically the location
                        1880  1722 ;  containing the first free rd and the list of rds).  Then read into local
                        1880  1723 ;  storage the first rd to display.  Check to see if this rd is on the free
                        1880  1724 ;  list, if it is then it will not be displayed.  Also if rd is permanently
```

```
                                                H 2
                                1880  1725 ;  allocated, it will not be displayed.  Otherwise it will be displayed.
                                1880  1726 ;
                 0000063D'EF    D4  1880  1727          clrl    rdt_size                        ; initialize field
                                1886  1728          getmem  rdt$w_size(r6),rdt_size,#2 ; size of rdt to read into virtual memory
                                1898  1729          retiferr                                ; return on error
                 00000639'EF    9F  189C  1730          pushab  rdt                             ; will contain virtual address for rdt
                 0000063D'EF    9F  18A2  1731          pushab  rdt_size                        ; size of rdt
              00000000'GF    02  FB  18A8  1732          calls   #2,g^lib$get_vm                 ; get memory for rdt
                                18AF  1733          retiferr                                ; return on error
                                18B3  1734          assume  rdt$c_length  eq  24
                                18B3  1735          getmem  -24(r6),@rdt,rdt_size           ; read rdt into storage
                                18C9  1736          retiferr                                ; return on error
59   00000639'EF    0000063D'EF    C1  18CD  1737          addl3   rdt_size,rdt,r9                 ; r9 => end address of rdt
     55    00000639'EF    18  C1  18D9  1738          addl3   #rdt$c_length,rdt,r5            ; base of rdt
                 5A    F8 A5  D0  18E1  1739          movl    rdt$l_maxrdidx(r5),r10          ; max index of rdt
                 58    F4 A5  D0  18E5  1740          movl    rdt$l_freerd(r5),r8             ; address of first free rdt entry
                    53    55  D0  18E9  1741          movl    r5,r3                           ; save base of rdt in r3
                                18EC  1742
                          54  D4  18EC  1743          clrl    r4                              ; initialize index counter to 0
                       59    55  D1  18EE  1744 10$:     cmpl    r5,r9                           ; check for end of rd list
                          2E  13  18F1  1745          beql    40$                             ; equal, end of list
                       58    65  D1  18F3  1746          cmpl    (r5),r8                         ; check to see if rd is free
                          1E  13  18F6  1747          beql    20$                             ; equal, hit a free rd
                 1C 04 A5    00  E1  18F8  1748          bbc     #rd$v_busy,4(r5),30$            ; not interested in perm. allocated rd's
                          57  D5  18FD  1749          tstl    r7                              ; cdt address specified
                          71  12  18FF  1750          bneq    50$                             ; yes, so check for a match on cdt
                          54  DD  1901  1751 15$:     pushl   r4                              ; index counter
                          55  DD  1903  1752          pushl   r5                              ; address of rd entry
              00001995'EF    01  FB  1905  1753          calls   #1,display_rd_entry             ; display one line of the summary page
                          54  D6  190C  1754          incl    r4                              ; increment index counter
                       55    08  C0  190E  1755          addl2   #8,r5                           ; advance to the next entry
                         DA  5A  F4  1911  1756          sobgeq  r10,10$                         ; get next rd
                          08  11  1914  1757          brb     40$                             ; end of list
                       58    65  D0  1916  1758 20$:     movl    (r5),r8                         ; next free rd
                          54  D6  1919  1759 30$:     incl    r4                              ; increment index counter
                       55    08  C0  191B  1760          addl2   #8,r5                           ; advance to the next entry
                         CD  5A  F4  191E  1761          sobgeq  r10,10$                         ; get next free rd
                                1921  1762 ;
                                1921  1763 ;  Now that all the busy rdt entries have been displayed, let's walk through
                                1921  1764 ;  the wait queue and display its contents.  These entries will not have a
                                1921  1765 ;  rspid since that is the reason they are in the wait queue.
                                1921  1766 ;
56   56    FFFFFFE8 8F    C1  1921  1767 40$:     addl3   #rdt$l_waitfl,r6,r6             ; start of wait queue
     55    00000641'EF    9E  1929  1768          movab   wait_cdrp,r5                    ; address to hold cdrp
           65    E8 A3  D0  1930  1769          movl    rdt$l_waitfl(r3),(r5)           ; first entry in wait queue
                    65    56  D1  1934  1770 41$:     cmpl    r6,(r5)                         ; end of queue?
                          22  13  1937  1771          beql    45$                             ; yes, so exit
                 FFFFFFFF 8F  DD  1939  1772          pushl   #-1                             ; not available rspid
                          55  DD  193F  1773          pushl   r5                              ; address of cdrp address
              00001995'EF    02  FB  1941  1774          calls   #2,display_rd_entry             ; display
                                1948  1775          getmem  a(r5),(r5),#4                   ; get next entry
                                1955  1776          retiferr                                ; return on error
                          D9  11  1959  1777          brb     41$                             ; loop
                                195B  1778 45$:
                 00000639'EF    9F  195B  1779          pushab  rdt                             ;
                 0000063D'EF    9F  1961  1780          pushab  rdt_size                        ; size to deallocate
              00000000'GF    02  FB  1967  1781          calls   #2,g^lib$free_vm                ; deallocate virtual memory
```

```
                                                                1  2
                     50   01    D0  196E  1782              movl    #1,r0                           ; success status
                                04  1971  1783              ret
                                     1972  1784
                                     1973  1785 50$:
                     58   65    D0  1973  1786              movl    (r5),r8                         ; cdrp address
                58   58   24    C1  1975  1787              addl3   #cdrp$l_cdt,r8,r8               ; point to cdt address
                                     1979  1788              getmem  (r8),r8                         ; cdt address
                     57   58    D1  1985  1789              cmpl    r8,r7                           ; match?
                          8F    12  1988  1790              bneq    30$                             ; not equal, don't display
                        FF74    31  198A  1791              brw     15$                             ; do display
                                     198D  1792
```

```
                                198D  1794          .sbttl      display_rd_entry --- display an entry in the response descriptor tab
                                198D  1795 ;---
                                198D  1796 ;
                                198D  1797 ;       display_rd_entry
                                198D  1798 ;
                                198D  1799 ;       This is a coroutine whose purpose is display each entry in the
                                198D  1800 ;       response descriptor table (RDT). A RDT is used to provide a
                                198D  1801 ;       match between a rspid and its CDRP.
                                198D  1802 ;
                                198D  1803 ;       Inputs:
                                198D  1804 ;
                                198D  1805 ;               4(ap) = rd entry in local storage
                                198D  1806 ;               8(ap) = index portion of rspid
                                198D  1807 ;
                                198D  1808 ;       Outputs:
                                198D  1809 ;
                                198D  1810 ;               A rd entry is displayed.
                                198D  1811 ;               All registers are preserved.
                                198D  1812 ;
                                198D  1813 ;---
                                198D  1814          .enabl  lsb
                                198D  1815 no_rspid:
67 6E 69 74 69 61 77 00'        198D  1816          .ascic /waiting/
                     07         198D
                                1995  1817 display_rd_entry::
                     0FFC       1995  1818          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                1997  1819
          54    04 AC    D0     1997  1820          movl    4(ap),r4                ; address of rd in local storage
             55    64    D0     199B  1821          movl    (r4),r5                 ; cdrp address
          56    55    24    C1 199E  1822          addl3   #cdrp$l_cdt,r5,r6       ; cdt address
                              19A2  1823          getmem  (r6),r6                 ; get pointer to cdt from cdrp
                        56    D5 19AE  1824          tstl    r6                      ; test for valid cdt
                        56    18 19B0  1825          bgeq    10$                     ; cdt is not valid
    57    00000008'EF       9E 19B2  1826          movab   cdt,r7                  ; address of local storage for cdt
                              19B9  1827          getmem  (r6),(r7),#cdt$c_length ; read into local storage
                              19CA  1828          retiferr                        ; return on error
                        57    DD 19CE  1829          pushl   r7                      ; address of cdt in local storage
             F998 CF    01    FB 19D0  1830          calls   #1,remote_node          ; find the remote node
                        5A    DD 19D5  1831          pushl   r10                     ; counted ascii string
                        57    DD 19D7  1832          pushl   r7                      ; address of cdt in local storage
             F90D CF    01    FB 19D9  1833          calls   #1,find_procname        ; find the local process name
                        52    DD 19DE  1834          pushl   r2                      ; address of local process name
                        53    DD 19E0  1835          pushl   r3                      ; length of name
                        56    DD 19E2  1836          pushl   r6                      ; cdt address
                        55    DD 19E4  1837 5$:       pushl   r5                      ; cdrp address
                  08 AC    D5 19E6  1838          tstl    8(ap)                   ; rspid available
                        27    19 19E9  1839          blss    20$                     ; branch if not available
             04 A4      00    B0 19EB  1840          movw    #0,4(r4)                ; zero the state field
    57    08 AC    04 A4    C9 19EF  1841          bisl3   4(r4),8(ap),r7          ; yields the rspid
                        57    DD 19F5  1842          pushl   r7                      ; rspid
                              19F7  1843          print   6,<!XL      !XL         !XL         !AD         !AC>
                  50    01    D0 1A04  1844 7$:       movl    #1,r0
                              04 1A07  1845          ret
                              1A08  1846
                  F6BD CF     9F 1A08  1847 10$:      pushab  null_string             ; empty string for node name
                              00 DD 1A0C  1848          pushl   #0                      ; length of remote node name
                              00 DD 1A0E  1849          pushl   #0                      ; cdt address
```

```
                D2  11  1A10  1850          brb     5$
                        1A12  1851
FF77 CF  9F     1A12  1852 20$:     pushab  no_rspid
                1A16  1853          print   5,^ !AC       !XL           !XL       !AD>
                DF  11  1A23  1854          brb     7$
                        1A25  1855
                        1A25  1856      .dsabl      lsb
```

```
                     1A25  1858          .sbttl      show_ports --- display all port descriptor tables (PDT)
                     1A25  1859 ;---
                     1A25  1860 ;
                     1A25  1861 ;       show_ports
                     1A25  1862 ;
                     1A25  1863 ;       This is the main routine whose purpose is to display the contents
                     1A25  1864 ;       of each port descriptor table (PDT). A PDT is used to store
                     1A25  1865 ;       scs entry addresses and port independent bookkeeping.  The first
                     1A25  1866 ;       page is a summary.
                     1A25  1867 ;
                     1A25  1868 ;       Inputs:
                     1A25  1869 ;
                     1A25  1870 ;               AP = pointer to TPARSE block
                     1A25  1871 ;
                     1A25  1872 ;       Outputs:
                     1A25  1873 ;
                     1A25  1874 ;               SCS data structures ( as mentioned above) are shown
                     1A25  1875 ;               All registers are preserved.
                     1A25  1876 ;
                     1A25  1877 ;---
                     1A25  1878          .enabl lsb
                     1A25  1879 show_ports::
             0FFC    1A25  1880          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                     1A27  1881 ;
                     1A27  1882 ;  Header information
                     1A27  1883 ;
                     1A27  1884          subhd   <VAXcluster data structures>            ; set heading
                     1A34  1885          skip    page
                     1A3B  1886          skip    1
                     1A44  1887          print   0,<!_!_  --- PDT Summary Page --->
                     1A51  1888          skip    1
                     1A5A  1889          print   0,< PDT Address       Type        Device          Driver Name>
                     1A67  1890          print   0,< -----------         ----        -------         ----------->
                     1A74  1891          skip    1
                     1A7D  1892 ;
                     1A7D  1893 ;  Read the contents of the pdt into local storage
                     1A7D  1894 ;
          54  00000525'EF   9E 1A7D  1895          movab   pdt,r4                          ; local storage for pdt
                     1A84  1896          getmem  @scs$gl_pdt,r5                  ; get address of 1st pdt
                     1A94  1897          retiferr                                ; return on error
               5B    55    D0 1A98  1898          movl    r5,r11                  ; save the pdt address
                     1A9B  1899 10$:     getmem  (r5),(r4),#pdt$c_length         ; read pdt into local storage
                     1AAC  1900          retiferr                                ; return on error
                     1AB0  1901
                     1AB0  1902 ;
                     1AB0  1903 ;  Get the driver name from the device data block (DDB).
                     1AB0  1904 ;
          52  00DC C4   D0 1AB0  1905          movl    pdt$l_ucb0(r4),r2               ; ucb address
          53  52    28   C1 1AB5  1906          addl3   #ucb$l_ddb,r2,r3                ; point to the ddb
                     1AB9  1907          getmem  (r3),r6                 ; ddb address
          58  56    24   C1 1AC5  1908          addl3   #ddb$b_drvnam_len,r6,r8         ; driver name length
                     1AC9  1909          getmem  (r8),r9                 ; read it in
                     1AD5  1910          retiferr                                ; return on error
               59    59   9A 1AD9  1911          movzbl  r9,r9                           ; zero the other fields
          57  000006C5'EF   9E 1ADC  1912          movab   driver_name,r7                  ; local storage for driver name
              58    56   24   C1 1AE3  1913          addl3   #ddb$t_drvname,r6,r8            ; point to driver name
                     1AE7  1914          getmem  1(r8),(r7),r9                   ; read into local storage
```

```
                                1AF5  1915          retiferr                                ; return on error
                          57    DD    1AF9  1916    pushl   r7                              ; address of driver name
                          59    DD    1AFB  1917    pushl   r9                              ; length of driver name
                                1AFD  1918 ;
                                1AFD  1919 ;  Put together the device name by pulling the unit number from the ucb and
                                1AFD  1920 ;  the device name from the device data block (DDB).
                                1AFD  1921 ;
      52    52    00000054 8F   C1    1AFD  1922          addl3   #ucb$w_unit,r2,r2              ; point to the unit field
                                1B05  1923          getmem  (r2),r2                         ; unit number
                          7E    52    3C    1B11  1924    movzwl  r2,-(sp)                        ; put on the stack
                    58    56    14    C1    1B14  1925    addl3   #ddb$b_name_len,r6,r8           ; point to length field
                                1B18  1926          getmem  (r8),r7                         ; read the field
                                1B24  1927          retiferr                                ; return on error
                          57    57    9A    1B28  1928    movzbl  r7,r7                           ; zero the other fields
                    58    56    14    C1    1B2B  1929    addl3   #ddb$t_name,r6,r8               ; point to name field
                56   000006D9'EF   9E    1B2F  1930    movab   device_name,r6                  ; local storage for name
                                1B36  1931          getmem  1(r8),(r6),r7                   ; read the name
                                1B44  1932          retiferr                                ; return on error
                          56    DD    1B48  1933    pushl   r6                              ; address of name
                          57    DD    1B4A  1934    pushl   r7                              ; push length on stack
                                1B4C  1935 ;
                                1B4C  1936 ;  Get the type of port (PA, PU, PE, PS).
                                1B4C  1937 ;
                      52    07 A4   9A    1B4C  1938    movzbl  pdt$b_pdt_type(r4),r2           ; port type
                53    E7AC   CF   9E    1B50  1939    movab   pdt_type,r3                     ; translation table
                 00000000'GF  16    1B55  1940    jsb     g^translate_address             ; translate constant to name
                          50    DD    1B5B  1941    pushl   r0                              ; address of ascii name
                          55    DD    1B5D  1942    pushl   r5                              ; pdt address
                                1B5F  1943          print   6,<  !XL                !AC             !AD!UW              !AD>
                                1B6C  1944
                          64    D5    1B6C  1945    tstl    pdt$l_flink(r4)                 ; is there another pdt
                          06    13    1B6E  1946    beql    20$                             ; equal, no
                    55    64    D0    1B70  1947    movl    pdt$l_flink(r4),r5              ; next pdt to display
                       FF25   31    1B73  1948    brw     10$                             ; loop for another pdt
                                1B76  1949 ;
                                1B76  1950 ;  Now that the summary page is complete, let's go on to display each pdt
                                1B76  1951 ;  in full.
                                1B76  1952 ;
                                1B76  1953          assume pdt$l_flink eq 0
                          5B    DD    1B76  1954 20$:  pushl   r11                             ; actual address of pdt
      00001B97'EF         02    FB    1B78  1955    calls   #2,display_pdt                  ; display this pdt
                                1B7F  1956          getmem  (r11),r11                       ; read link field
                                1B8B  1957          retiferr                                ; return on error
                          5B    D5    1B8F  1958    tstl    r11                             ; another pdt
                          E3    12    1B91  1959    bneq    20$                             ; not equal, display next pdt
                                1B93  1960
                    50    01    D0    1B93  1961    movl    #1,r0                           ; return with success
                                04    1B96  1962    ret
                                1B97  1963
```

```
                        1B97  1965      .sbttl      display_pdt --- display a port descriptor table
                        1B97  1966 ;---
                        1B97  1967 ;
                        1B97  1968 ;       display_pdt
                        1B97  1969 ;
                        1B97  1970 ;       This is a coroutine whose purpose is display each port
                        1B97  1971 ;       descriptor table (PDT).
                        1B97  1972 ;
                        1B97  1973 ;       Inputs:
                        1B97  1974 ;
                        1B97  1975 ;               4(ap) = actual address of pdt
                        1B97  1976 ;
                        1B97  1977 ;       Outputs:
                        1B97  1978 ;
                        1B97  1979 ;               PDT is displayed.
                        1B97  1980 ;               All registers are preserved.
                        1B97  1981 ;
                        1B97  1982 ;---
                        1B97  1983          .enabl  lsb
                        1B97  1984 display_pdt::
                OFFC    1B97  1985          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                        1B99  1986
                        1B99  1987          skip    page
                        1BA0  1988          skip    1
              04 AC  DD 1BA9  1989          pushl   4(ap)
                        1BAC  1990          print   0,<!_!_  --- Port Descriptor Table (PDT) !XL --->
                        1BB9  1991          skip    1
                        1BC2  1992
    55  00000525'EF   DE 1BC2  1993          moval   pdt,r5                          ; local storage for pdt
                        1BC9  1994          getmem  @4(ap),(r5),#pdt$c_length       ; read it in
                        1BDB  1995          retiferr                                ; return on error
                        1BDF  1996 ;
                        1BDF  1997 ;  Translate port type
                        1BDF  1998 ;
      52    07 A5   9A 1BDF  1999          movzbl  pdt$b_pdt_type(r5),r2
   53    E719 CF   9E 1BE3  2000          movab   pdt_type,r3
    00000000'GF   16 1BE8  2001          jsb     g^translate_address             ; get the ascii name for the
                        1BEE  2002                                                  ; port type
            50  DD 1BEE  2003          pushl   r0                              ; counted ascii string
      7E    07 A5   9A 1BF0  2004          movzbl  pdt$b_pdt_type(r5),-(sp)        ; port type
                        1BF4  2005          print   2,<Type: !XB !AC>
                        1C01  2006 ;
                        1C01  2007 ;  Translate port characteristics
                        1C01  2008 ;
                        1C01  2009          alloc   80                              ; output buffer
      7E    04 A5   3C 1C10  2010          movzwl  pdt$w_portchar(r5),-(sp)        ; port characteristics
        E710 CF   9F 1C14  2011          pushab  port_char                       ; bit definition table
00000000'EF    02 FB 1C18  2012          calls   #2,translate_bits               ; translate bits to names
            5E  DD 1C1F  2013          pushl   sp                              ; address of string descriptor
      7E    04 A5   3C 1C21  2014          movzwl  pdt$w_portchar(r5),-(sp)        ; port characteristics
                        1C25  2015          print   2,<Characteristics: !XW !AS>
  5E    00000050 8F   C0 1C32  2016          addl2   #80,sp                          ; clean up the stack
                        1C39  2017          skip    1
                        1C42  2018 ;
                        1C42  2019 ;  Display the rest of the pdt
                        1C42  2020 ;
                        1C42  2021          make_symbol     UCB, pdt$l_ucb0(r5)
```

```
                     1C59  2022          make_symbol     ADP, pdt$l_adp(r5)
                     1C70  2023
                     1C70  2024          print_columns   -
                     1C70  2025                  (r5),4(ap),-
                     1C70  2026                  pdt_col_1,pdt_col_2,pdt_col_3   ; display
50   01   D0     1C8E  2027          movl    #1,r0                               ; return with success
               04    1C91  2028          ret
                     1C92  2029          .dsabl  lsb
```

```
                        1C92  2031      .sbttl      pdt_byaddr --- display the pdt requested by the user
                        1C92  2032  ;---
                        1C92  2033  ;
                        1C92  2034  ;       pdt_byaddr
                        1C92  2035  ;
                        1C92  2036  ;       This is a routine whose purpose is display a port
                        1C92  2037  ;       descriptor table (PDT) which the user has requested by using
                        1C92  2038  ;       the /ADDR qualifier and a valid address of a pdt.
                        1C92  2039  ;
                        1C92  2040  ;       Inputs:
                        1C92  2041  ;
                        1C92  2042  ;               AP = TPARSE block (TPA$L_NUMBER contains the address)
                        1C92  2043  ;
                        1C92  2044  ;       Outputs:
                        1C92  2045  ;
                        1C92  2046  ;               The requested PDT is displayed if a valid address is specified.
                        1C92  2047  ;               Otherwise an informational message is sent to say invalid pdt
                        1C92  2048  ;               address.
                        1C92  2049  ;               All registers are preserved.
                        1C92  2050  ;
                        1C92  2051  ;---
                        1C92  2052          .enabl  lsb
                        1C92  2053  pdt_byaddr::
              0FFC      1C92  2054          .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
     57   1C AC    D0   1C94  2055          movl    tpa$l_number(ap),r7             ; get address of pdt
00001CC0'EF     16   1C98  2056          jsb     verify_pdt                      ; is this a pdt
        1B 50    E9   1C9E  2057          blbc    r0,10$                          ; clear, not a pdt
                        1CA1  2058  ;
                        1CA1  2059  ;  Now that we are through the validation phase the rest is trivial.
                        1CA1  2060  ;
                        1CA1  2061          subhd   <VAXcluster data structures>            ; set heading
                        1CAE  2062          skip    page
           57   DD      1CB5  2063          pushl   r7                              ; pass the actual address
 FEDB CF    01   FB      1CB7  2064          calls   #1, display_pdt                 ; display it
                        1CBC  2065  10$:
      50    01   D0      1CBC  2066          movl    #1,r0
                  04    1CBF  2067          ret
                        1CC0  2068
                        1CC0  2069  verify_pdt:
           57   D5      1CC0  2070          tstl    r7                              ; check for 80000000 address
           2B   18      1CC2  2071          bgeq    900$                            ; not valid
   58   57   0A   C1    1CC4  2072          addl3   #pdt$b_type,r7,r8               ; point at the type field
                        1CC8  2073          getmem  (r8),r8                         ; attempt to read
                        1CD4  2074          retiferr                                ; return on error
     60 8F    58   91    1CD8  2075          cmpb    r8,#dyn$c_scs                   ; check for the right type
              0E   12    1CDC  2076          bneq    900$                            ; not equal, error
58   58   F8 8F   78    1CDE  2077          ashl    #-8,r8,r8                       ; point at subtype
        05    58   91    1CE3  2078          cmpb    r8,#dyn$c_scs_pdt               ; check for correct subtype
              04   12    1CE6  2079          bneq    900$                            ; not equal, invalid address
      50    01   D0      1CE8  2080          movl    #1,r0                           ; valid pdt
                  05    1CEB  2081          rsb
                        1CEC  2082
           57   DD      1CEC  2083  900$:   pushl   r7                              ; invalid pdt address
                        1CEE  2084          type    1,<!XL is not the address of a PDT>
      50    00   D0      1D36  2085          movl    #0,r0                           ; invalid pdt
                  05    1D39  2086          rsb
                        1D3A  2087          .dsabl  lsb
```

```
                        1D3A  2089      .sbttl      port descriptor tables & action routines
                        1D3A  2090 ;
                        1D3A  2091 ;    PRINT_COLUMNS table for PDT displays
                        1D3A  2092 ;
                        1D3A  2093
                        1D3A  2094
                        1D3A  2095 pdt_col_1:
                        1D3A  2096          column_list     -
                        1D3A  2097                  pdt$, 20, 8, 2, < -
                        1D3A  2098                  <<Msg Header Size>,l_msghdrsz,ul>,-
                        1D3A  2099                  <<Max Xfer Bcnt>,l_maxbcnt,xl>,-
                        1D3A  2100                  <<DG Header Size>,l_dgovrhd,ul>,-
                        1D3A  2101                  <<Poller Sweep>,l_pollsweep,ul>,-
                        1D3A  2102                  <<Fork Block W.Q.S.>,l_waitqfl,q2>,-
                        1D3A  2103                  <<UCB Address>,l_ucb0,xl>,-
                        1D3A  2104                  <<ADP Address>,l_adp,xl>,-
                        1D3A  2105                  <<Accept>,l_accept,xl>,-
                        1D3A  2106                  <<Alloc_Dg_Buf>,l_allocdg,xl>,-
                        1D3A  2107                  <<Alloc_Msg_Buf>,l_allocmsg,xl>,-
                        1D3A  2108                  <<Dealloc_Msg_Buf>,l_deallomsg,xl>,-
                        1D3A  2109                  <<Dealloc_Msg_Buf_Reg>,l_dealrgmsg,xl>,-
                        1D3A  2110                  >
                        1E0A  2111 pdt_col_2:
                        1E0A  2112          column_list     -
                        1E0A  2113                  pdt$, 15, 8, 2, < -
                        1E0A  2114                  <<Connect>,l_connect,xl>,-
                        1E0A  2115                  <<Dealloc_Dg_Buf>,l_deallocdg,xl>,-
                        1E0A  2116                  <<Disconnect>,l_dconnect,xl>,-
                        1E0A  2117                  <<Unmap>,l_unmap,xl>,-
                        1E0A  2118                  <<Map>,l_map,xl>,-
                        1E0A  2119                  <<Map_Bypass>,l_mapbypass,xl>,-
                        1E0A  2120                  <<Map_Irp>,l_mapirp,xl>,-
                        1E0A  2121                  <<Map_Irp_Bypass>,l_mapirpbyp,xl>,-
                        1E0A  2122                  <<Queue_Dg_Buf>,l_queuedg,xl>,-
                        1E0A  2123                  <<Queue_Mult_Dgs>,l_queuemdgs,xl>,-
                        1E0A  2124                  <<Recycl_Msg_Buf>,l_rclmsgbuf,xl>,-
                        1E0A  2125                  <<Reject>,l_reject,xl>,-
                        1E0A  2126                  >
                        1EDA  2127 pdt_col_3:
                        1EDA  2128          column_list     -
                        1EDA  2129                  pdt$, 17, 8, 0, < -
                        1EDA  2130                  <<Recycln_Msg_Buf>,l_rchmsgbuf,xl>,-
                        1EDA  2131                  <<Request_Data>,l_reqdata,xl>,-
                        1EDA  2132                  <<Send_Data>,l_senddata,xl>,-
                        1EDA  2133                  <<Send_Dg_Buf>,l_senddg,xl>,-
                        1EDA  2134                  <<Send_Msg_Buf>,l_sendmsg,xl>,-
                        1EDA  2135                  <<Send_Cnt_Msg_Buf>,l_sndcntmsg,xl>,-
                        1EDA  2136                  <<Read_Count>,l_readcount,xl>,-
                        1EDA  2137                  <<Rls_Read_Count>,l_rlscount,xl>,-
                        1EDA  2138                  <<Mreset>,l_mreset,xl>,-
                        1EDA  2139                  <<Mstart>,l_mstart,xl>,-
                        1EDA  2140                  <<Stop_Vcs>,l_stop_vcs,xl>,-
                        1EDA  2141                  <<Send_Dg_Reg>,l_sendrgdg,xl>,-
                        1EDA  2142                  >
                        1FAA  2143      .end
```

```
$$$                             = 000002F5 R      04
$$.TMP1                         = 00000001
$$.TMP2                         = 000000EF
$$T2                            = 00000005
ADD_SYMBOL                        ********    X   03
ARGS                            = 00000001
CDL                               00000000 R      02
CDL$L_FREECDT                   = FFFFFFF4
CDL$W_MAXCONIDX                 = FFFFFFF0
CDL$W_SIZE                      = FFFFFFF8
CDL_SIZE                          00000004 R      02
CDRP$L_CDT                      = 00000024
CDT                               00000008 R      02
CDT$B_RSTATION                  = 00000020
CDT$B_TYPE                      = 0000000A
CDT$C_ACCP_PEND                 = 00000002
CDT$C_ACCP_SENT                 = 0000000A
CDT$C_CLOSED                    = 00000000
CDT$C_CON_ACK                   = 00000008
CDT$C_CON_PEND                  = 00000001
CDT$C_CON_REC                   = 00000009
CDT$C_CON_SENT                  = 00000007
CDT$C_CR_PEND                   = 00000005
CDT$C_DCR_PEND                  = 00000006
CDT$C_DISC_ACK                  = 00000003
CDT$C_DISC_MTCH                 = 00000006
CDT$C_DISC_PEND                 = 00000004
CDT$C_DISC_REC                  = 00000004
CDT$C_DISC_SENT                 = 00000005
CDT$C_LENGTH                    = 000000A0
CDT$C_LISTEN                    = 00000001
CDT$C_OPEN                      = 00000002
CDT$C_REJ_PEND                  = 00000003
CDT$C_REJ_SENT                  = 00000008
CDT$C_VC_FAIL                   = 0000000C
CDT$L_AUXSTRUC                  = 0000005C
CDT$L_BYTMAPD                   = 00000094
CDT$L_BYTREQD                   = 00000090
CDT$L_BYTSENT                   = 00000088
CDT$L_CONDAT                    = 00000058
CDT$L_CRWAITQFL                 = 00000038
CDT$L_DGDISCARD                 = 00000078
CDT$L_DGRCVD                    = 00000074
CDT$L_DGSENT                    = 00000070
CDT$L_ERRADDR                   = 0000000C
CDT$L_LCONID                    = 00000018
CDT$L_LPROCNAM                  = 00000054
CDT$L_MSGRCVD                   = 00000080
CDT$L_MSGSENT                   = 0000007C
CDT$L_PB                        = 0000001C
CDT$L_PDT                       = 00000010
CDT$L_RCONID                    = 00000014
CDT$L_REQDATS                   = 0000008C
CDT$L_RPROCNAM                  = 00000050
CDT$L_SCSMSG                    = 0000002C
CDT$L_SNDDATS                   = 00000084
CDT$L_WAITQFL                   = 00000030
CDT$W_BLKSTATE                  = 0000002A
CDT$W_INITLREC                  = 00000048
CDT$W_MINREC                    = 00000044
CDT$W_PENDREC                   = 00000046
CDT$W_QBDT_CNT                  = 0000009A
CDT$W_QCR_CNT                   = 00000098
CDT$W_REASON                    = 00000026
CDT$W_REC                       = 00000042
CDT$W_SEND                      = 00000040
CDT$W_STATE                     = 00000028
CDT_6BYTES                        000017D7 R      03
CDT_BLKSTATE                      000002C8 R      03
CDT_BYADDR                        0000152E RG     03
CDT_COL_1                         000015C9 R      03
CDT_COL_2                         00001689 R      03
CDT_COL_3                         00001739 R      03
CDT_FAO_6BYTES                    000017C9 R      03
CDT_SPCFY                         000006F9 RG     02
CDT_STATE                         00000258 R      03
CLU$GL_CLUB                       ********    X   03
CLUB                              000000A8 R      02
CLUB$B_CLUFCB                   = 0000010C
CLUB$B_CUR_CODE                 = 00000058
CLUB$B_CUR_PHASE                = 00000059
CLUB$B_FSYSID                   = 00000026
CLUB$B_LST_CODE                 = 00000044
CLUB$C_LENGTH                   = 000001A8
CLUB$L_CLUDCB                   = 000000B4
CLUB$L_CSBQFL                   = 00000000
CLUB$L_CSPFL                    = 00000088
CLUB$L_CUR_COORD                = 0000004C
CLUB$L_CUR_XTN                  = 00000048
CLUB$L_FLAGS                    = 0000001C
CLUB$L_FMERIT                   = 000000A8
CLUB$L_FOREIGN_CLUSTER          = 000000C8
CLUB$L_LST_COORD                = 00000038
CLUB$L_LST_XTN                  = 00000034
CLUB$L_MAX_XTN                  = 00000068
CLUB$L_RETRYCNT                 = 0000006C
CLUB$L_TQE                      = 00000084
CLUB$Q_CUR_TIME                 = 00000050
CLUB$Q_FTIME                    = 0000002C
CLUB$Q_LST_TIME                 = 0000003C
CLUB$T_QDNAME                   = 000000B8
CLUB$V_ADJ_QUORUM               = 0000001B
CLUB$V_BACKOUT                  = 00000014
CLUB$V_CLUSTER                  = 00000000
CLUB$V_FKB_BUSY                 = 00000010
CLUB$V_INIT                     = 00000013
CLUB$V_LOST_CNX                 = 00000017
CLUB$V_NO_FORM                  = 00000012
CLUB$V_QF_ACTIVE                = 00000001
CLUB$V_QF_DYNVOTE               = 0000001E
CLUB$V_QF_FAILED_NODE           = 00000018
CLUB$V_QF_NEWVOTE               = 0000001A
CLUB$V_QF_VOTE                  = 00000019
CLUB$V_QUORUM                   = 0000001C
```

```
CLUB$V_SHUTDOWN                 = 00000002
CLUB$V_STS_PH0                  = 00000009
CLUB$V_STS_PH1                  = 0000000B
CLUB$V_STS_PH1B                 = 0000000A
CLUB$V_STS_PH2                  = 0000000C
CLUB$V_STS_PPHASE               = 00000008
CLUB$V_TRANSITION               = 0000001D
CLUB$V_UNLOCK                   = 00000011
CLUB$W_MEMSEQ                   = 000000AC
CLUB$W_MSGCNT                   = 0000005A
CLUB$W_NEXT_CSID                = 00000064
CLUB$W_NODES                    = 00000024
CLUB$W_QDVOTES                  = 000000AE
CLUB$W_QUORUM                   = 00000020
CLUB$W_VOTES                    = 00000022
CLUB_2WORDS                       00000653 R      03
CLUB_6BYTES                       00000860 R      03
CLUB_COL_1                        00000664 R      03
CLUB_COL_2                        00000754 R      03
CLUB_FAO_6BYTES                   00000641 R      03
CLUB_FLAGS                        00000148 R      03
CLUB_SUMMARY                      00000000 R      03
CLUDCB                            000002FC R      02
CLUDCB$B_COUNTER                = 00000024
CLUDCB$C_LENGTH                 = 00000229
CLUDCB$L_ACT_COUNT              = 00000018
CLUDCB$L_IRP                    = 00000010
CLUDCB$L_QFLBN                  = 0000001C
CLUDCB$L_TQE                    = 00000014
CLUDCB$L_UCB                    = 0000000C
CLUDCB$V_QF_CSPACK              = 00000004
CLUDCB$V_QF_ERROR               = 00000003
CLUDCB$V_QF_RIP                 = 00000001
CLUDCB$V_QF_TIM                 = 00000000
CLUDCB$V_QF_WIP                 = 00000002
CLUDCB$V_QS_ACTIVE              = 00000002
CLUDCB$V_QS_CLUSTER             = 00000003
CLUDCB$V_QS_NOT_READY           = 00000000
CLUDCB$V_QS_READY               = 00000001
CLUDCB$V_QS_VOTE                = 00000004
CLUDCB$W_FLAGS                  = 00000022
CLUDCB$W_STATE                  = 00000020
CLUDCB_FLAGS                      00000228 R      03
CLUDCB_STATE                      000001F8 R      03
CLUFCB$L_ID                     = 0000001C
CLUFCB$L_STATUS                 = 00000020
CLUFCB$L_STEP                   = 00000018
CLUFCB$L_SYNC_CSB               = 00000024
CLUFCB$V_ACTIVE                 = 00000000
CLUFCB$V_FKB_BUSY               = 00000003
CLUFCB$V_PENDING                = 00000001
CLUFCB$V_SYNC_NODE              = 00000002
CLUSTER_SUMMARY                   0000043F R      03
CMND_BUFFER                       ******** X    03
CMND_DESCR                        ******** X    03
COLM$K_FAO_AC                   = 00000000
COLM$K_FAO_AS                   = 00000001
COLM$K_FAO_Q2                   = 00000011
COLM$K_FAO_UB                   = 00000005
COLM$K_FAO_UL                   = 0000000F
COLM$K_FAO_UW                   = 0000000A
COLM$K_FAO_XB                   = 00000003
COLM$K_FAO_XL                   = 0000000D
COLM$K_FAO_XW                   = 00000008
COLM$K_LENGTH                   = 00000010
COUNT_PATHS                       0000106B R      03
CSB                               00000250 R      02
CSB$B_ECOLVL                    = 00000040
CSB$B_REF_CNT                   = 0000006C
CSB$B_REMACKLIM                 = 00000033
CSB$B_STATE                     = 00000043
CSB$B_UNACKEDMSGS               = 00000032
CSB$B_VERNUM                    = 00000041
CSB$C_LENGTH                    = 000000AC
CSB$K_ACCEPT                    = 00000006
CSB$K_CONNECT                   = 00000005
CSB$K_DEAD                      = 0000000A
CSB$K_DISCONNECT                = 00000007
CSB$K_LOCAL                     = 0000000B
CSB$K_NEW                       = 00000004
CSB$K_OPEN                      = 00000001
CSB$K_REACCEPT                  = 00000008
CSB$K_RECONNECT                 = 00000003
CSB$K_STATUS                    = 00000002
CSB$K_WAIT                      = 00000009
CSB$L_CDT                       = 0000000C
CSB$L_CSID                      = 0000004C
CSB$L_CURRCDRP                  = 00000034
CSB$L_PARTNERQFL                = 00000058
CSB$L_PDT                       = 00000010
CSB$L_RESENDQFL                 = 0000001C
CSB$L_SB                        = 00000068
CSB$L_SENTQFL                   = 00000014
CSB$L_STATUS                    = 00000060
CSB$L_SYSQFL                    = 00000000
CSB$L_TIMEOUT                   = 00000048
CSB$L_TQE                       = 00000044
CSB$Q_REFTIME                   = 00000074
CSB$Q_SWINCARN                  = 00000038
CSB$V_CLUSTER                   = 00000008
CSB$V_LOCAL                     = 00000018
CSB$V_LOCKED                    = 00000010
CSB$V_LONG_BREAK                = 00000000
CSB$V_MEMBER                    = 00000001
CSB$V_QF_ACTIVE                 = 00000009
CSB$V_QF_SAME                   = 00000003
CSB$V_REMOVED                   = 00000002
CSB$V_SELECTED                  = 00000011
CSB$V_SEND_STATUS               = 0000001A
CSB$V_SHUTDOWN                  = 0000000A
CSB$V_STATUS_RCVD               = 00000019
CSB$W_ACKRSEQNM                 = 00000030
CSB$W_LCKDIRWT                  = 00000054
CSB$W_QDVOTES                   = 00000056
```

```
CSB$W_QUORUM                =  00000052
CSB$W_RCVDSEQNM             =  0000002E
CSB$W_SENDSEQNM             =  0000002C
CSB$W_VOTES                 =  00000050
CSBQUOR_VOTES                  00000E9F  R      03
CSB_2BYTES                     00000CDF  R      03
CSB_2WORDS                     00000CCF  R      03
CSB_COL_1                      00000CEF  R      03
CSB_COL_2                      00000D7F  R      03
CSB_COL_3                      00000E0F  R      03
CSB_STATES                     00000058  R      03
CSB_STATUS                     000000B8  R      03
CSB_SUMMARY                    00000028  R      03
CSID                           000006F5  RG     02
CURR_DATE                      000008F5  R      03
CURR_TIME                      000008FE  R      03
DATE_ROUTINE                   0000087E  R      03
DCB_COL_1                      00000AF2  R      03
DCB_COL_2                      00000B32  R      03
DDB$B_DRVNAM_LEN            =  00000024
DDB$B_NAME_LEN              =  00000014
DDB$T_DRVNAME               =  00000024
DDB$T_NAME                  =  00000014
DEVICE_NAME                    000006D9  R      02
DIRECTORY                      00000609  R      02
DISPLAY_CDT                    000013A9  RG     03
DISPLAY_CLUB                   000005A8  R      03
DISPLAY_CLUDCB                 000009D6  R      03
DISPLAY_CLUFCB                 0000093D  R      03
DISPLAY_CSB                    00000B72  R      03
DISPLAY_PDT                    00001B97  RG     03
DISPLAY_RD_ENTRY               00001995  RG     03
DISPLAY_SB_PBS                 00001091  R      03
DISPLAY_SUMLINE                00001281  R      03
DONE                           000005A4  R      03
DRIVER_NAME                    000006C5  R      02
DYN$C_SCS                   =  00000060
DYN$C_SCS_CDT               =  00000002
DYN$C_SCS_PDT               =  00000005
ECO_VERS                       00000EBB  R      03
FAB$L_STV                      ********    X    03
FCB_COL_1                      00000AA2  R      03
FCB_COL_2                      00000AD2  R      03
FCB_STATUS                     00000120  R      03
FIND_PROCNAME                  000012EB  RG     03
FREE_CDT_LIST                  00001258  R      03
GETMEM                         ********      X  03
LIB$FREE_VM                    ********      X  03
LIB$GET_VM                     ********      X  03
LIB$SIGNAL                     ********      X  03
LINE_COUNT                     ********      X  03
LOCATE_CSB                     000003E5  R      03
LOOP                           0000050E  R      03
MAKE_CSB_SYMBOLS               00000BEB  R      03
MSG$_SUCCESS                   ********    X    03
NEW_PAGE                       ********    X    03
NODE                           000006A5  R      02
NOTRANS                        00000C67  R      03
NO_RSPID                       0000198D  R      03
NUL_STRING                     000010C9  R      03
OUTPUT                         ********    X    03
PAGE_SIZE                      ********    X    03
PB$L_FLINK                  =  00000000
PB$L_SBLINK                 =  00000030
PDT                            00000525  R      02
PDT$B_PDT_TYPE              =  00000007
PDT$B_TYPE                  =  0000000A
PDT$C_LENGTH                =  000000E4
PDT$C_PA                    =  00000001
PDT$C_PE                    =  00000003
PDT$C_PS                    =  00000004
PDT$C_PU                    =  00000002
PDT$L_ACCEPT                =  0000000C
PDT$L_ADP                   =  000000E0
PDT$L_ALLOCDG               =  00000010
PDT$L_ALLOCMSG              =  00000014
PDT$L_CONNECT               =  00000018
PDT$L_DCONNECT              =  00000028
PDT$L_DEALLOCDG             =  0000001C
PDT$L_DEALLOMSG             =  00000020
PDT$L_DEALRGMSG             =  00000024
PDT$L_DGOVRHD               =  00000088
PDT$L_FLINK                 =  00000000
PDT$L_MAP                   =  0000002C
PDT$L_MAPBYPASS             =  00000030
PDT$L_MAPIRP                =  00000034
PDT$L_MAPIRPBYP             =  00000038
PDT$L_MAXBCNT               =  000000BC
PDT$L_MRESET                =  00000070
PDT$L_MSGHDRSZ              =  000000B4
PDT$L_MSTART                =  00000074
PDT$L_POLLSWEEP             =  000000D8
PDT$L_QUEUEDG               =  0000003C
PDT$L_QUEUEMDGS             =  00000040
PDT$L_RCHMSGBUF             =  00000044
PDT$L_RCLMSGBUF             =  00000048
PDT$L_READCOUNT             =  00000068
PDT$L_REJECT                =  0000004C
PDT$L_REQDATA               =  00000050
PDT$L_RLSCOUNT              =  0000006C
PDT$L_SENDDATA              =  00000054
PDT$L_SENDDG                =  00000058
PDT$L_SENDMSG               =  0000005C
PDT$L_SENDRGDG              =  0000007C
PDT$L_SNDCNTMSG             =  00000060
PDT$L_STOP_VCS              =  00000080
PDT$L_UCBO                  =  000000DC
PDT$L_UNMAP                 =  00000064
PDT$L_WAITQFL               =  000000AC
PDT$V_SNGLHOST              =  00000000
PDT$W_PORTCHAR              =  00000004
PDT_BYADDR                     00001C92  RG     03
PDT_COL_1                      00001D3A  R      03
PDT_COL_2                      00001E0A  R      03
```

```
PDT_COL_3                       00001EDA R     03
PDT_TYPE                        00000300 R     03
PORT_CHAR                       00000328 R     03
PRINT                           ******** X     03
PRINT_COLUMNS                   ******** X     03
PRINT_COLUMN_VALUE              ******** X     03
PROCNAME                        000006B5 R     02
QUOR_VOTE                       00000844 R     03
RAB$C_RBF                       ******** X     03
RAB$W_RSZ                       ******** X     03
RD$V_BUSY                     = 00000000
RDT                             00000639 R     02
RDT$C_LENGTH                  = 00000018
RDT$L_FREERD                  = FFFFFFF4
RDT$L_MAXRDIDX                = FFFFFFF8
RDT$L_WAITFL                  = FFFFFFE8
RDT$W_SIZE                    = FFFFFFF0
RDT_SIZE                        0000063D R     02
REMOTE_NODE                     0000136D R     03
SB$B_SYSTEMID                 = 00000018
SB$C_LENGTH                   = 00000060
SB$L_FLINK                    = 00000000
SB$L_PBFL                     = 0000000C
SB$S_NODENAME                 = 00000010
SB$T_NODENAME                 = 00000044
SB$T_SWTYPE                   = 00000024
SBLOCK                          00000645 R     02
SCS$GL_CDL                      ******** X     03
SCS$GL_PDT                      ******** X     03
SCS$GL_RDT                      ******** X     03
SCS$GQ_CONFIG                   ******** X     03
SCS$GQ_DIRECT                   ******** X     03
SCS_SUMMARY                     00000F06 R     03
SDIR$B_PROCINF                = 0000001C
SDIR$B_PROCNAM                = 0000000C
SDIR$C_LENGTH                 = 00000030
SDIR$L_CONID                  = 0000002C
SDIR$L_FLINK                  = 00000000
SET_HEADING                     ******** X     03
SHOW_CLUSTER                    00000338 RG    03
SHOW_CONNECTIONS                000010CA RG    03
SHOW_PORTS                      00001A25 RG    03
SHOW_RSPID                      000017F4 RG    03
SHOW_SCS                        00000ED7 RG    03
SHOW_SYSTEM_BLOCK               ******** X     03
SKIP_LINES                      ******** X     03
STATE_TRANSLATE                 000012CA R     03
SYS$ASCTIM                      ******** GX    03
SYS$FAO                         ******** X     03
SYS$PUT                         ******** GX    03
TIME_ROUTINE                    000008B6 R     03
TIM_BUFFER                      000006ED R     02
TPA$L_NUMBER                  = 0000001C
TRANSLATE_ADDRESS               ******** X     03
TRANSLATE_BITS                  ******** X     03
TRANS_BYTE                      0000092B R     03
TRANS_LONG                      00000907 R     03
TRANS_WORD                      00000919 R     03
UCB$L_DDB                     = 00000028
UCB$W_UNIT                    = 00000054
VERIFY_CDT                      0000154F R     03
VERIFY_PDT                      00001CC0 R     03
WAIT_CORP                       00000641 R     02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
SDADATA	000006FD (1789.)	02 (2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC BYTE
CLUSTER	00001FAA (8106.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG
LITERALS	0000149C (5276.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.06	00:00:00.77
Command processing	136	00:00:00.42	00:00:04.62
Pass 1	649	00:00:20.19	00:01:09.47
Symbol table sort	0	00:00:01.73	00:00:05.20
Pass 2	418	00:00:05.87	00:00:21.36
Symbol table output	41	00:00:00.24	00:00:00.71
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1283	00:00:28.53	00:01:42.16

The working set limit was 2100 pages.
189601 bytes (371 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1388 non-local and 314 local symbols.
2143 source lines were read in Pass 1, producing 70 object records in Pass 2.
50 pages of virtual memory were used to define 46 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SDA.OBJ]SDALIB.MLB;1	17
_255DUA28:[SYS.OBJ]LIB.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	42

1632 GETS were required to define 42 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:CLUSTER/OBJ=OBJ$:CLUSTER MSRC$:CLUSTER/UPDATE=(ENH$:CLUSTER)+EXECML$/LIB+LIB$:SDALIB/LIB
```

XCASE
LIS

SDADEF
SDL

RMSDEF
MAR

UNPUS
LIS

SDATEST
MAP

SDA

SDA
MAP

DCLDEF
MAR

SCSDEF
MAR

TPR
LIS

XTAB
LIS

IPROBE
LIS

SYSDEF
MAR

IMGDEF
MAR

NETDEF
MAR

VAXOPS
REQ

MACROS
MAR

CLUSTER
LIS

DEVICE
LIS

CRASH
LIS

COMMANDS
LIS

DECODE
LIS